Mise en oeuvre d'une plateforme de gestion et de dissémination des connaissances pour des réseaux autonomiques

Sami Souihi

UNIVERSITÉ PARIS-EST

École doctorale Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

T H È S E

Pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-EST

Specialite : Génie Informatique et Réseaux

Présentée par

Sami SOUIHI

Mise en oeuvre d'une plateforme de gestion et de dissémination des connaissances pour des réseaux autonomiques

Soutenue publiquement le 03 décembre 2013

devant le jury composé de :

Rapporteurs :			
	Dominique GAÏTI	-	Université de Technologie de Troyes
	Ken CHEN	-	Université de Paris 13
Directeurs :			
	Abdelhamid MELLOUK	-	Université Paris-Est Creteil
	Co-encadrant : Said HOCEINI	-	Université Paris-Est Creteil
Examinateurs :			
	Guy PUJOLLE	-	Université Paris 6
	Ana Rosa CAVALI	-	Inst. Nat. Télécommunications
	Prosper CHEMOUIL	-	Orange Labs

Remerciements

Je tiens à remercier en premier lieu mon directeur de thèse, le Professeur Abdelhamid MELLOUK et mon co-encadrant, Dr. Said HOCEINI, pour la confiance et la liberté qu'ils m'ont accordées et pour m'avoir guidé dans les moments de doute.

Ce travail n'aurait pu aboutir sans l'aide de nombreuses personnes. Que me pardonnent ici celles que j'oublie, mais j'adresse une pensée particulière à Mohamed, Julien, Hai anh, Benjamin, Walid, Brice, Nadjib, Karim ainsi qu'à tous les membres du laboratoire LiSSi.

Un immense merci à Sondes, ma douce femme, qui m'a supporté et aidé durant les longues veillées qui précèdent les "Deadlines".

Mes dernières pensées iront vers ma famille, et surtout mes parents, qui m'auront permis de poursuivre mes études jusqu'à aujourd'hui.

Liste des publications

Revues internationales avec comité de lecture :

Souihi, S. ; Hoceini, S. ; Mellouk, A. ; Augustin B. ; Ait Saadi N., "SMILAY : An information flow management framework for microgrid applications", IEEE Communications Magazine, 51 (1), pp. 120-126, 2013.

Conférences internationales avec comité de lecture et actes :

Souihi, S. ; Perez J. ; Hoceini, S. ; Mellouk, A., "A robust, adaptative and hierarchical knowledge dissemination architecture", 2013 IEEE Global Communications Conference, Globecom 2013, 9-13 décembre 2013, Atlanta, GA, USA.

Souihi, S. ; Hoceini, S. ; Mellouk, A. ; Ait Saadi N., "A hierarchical and multi-criteria knowledge dissemination in autonomic networks", 2012 IEEE Global Communications Conference, Globecom 2012, 3-7 décembre 2012, Anaheim, CA, USA.

Souihi, S. ; Hoceini, S. ; Mellouk, A., "A multi-criteria master nodes selection mechanism for knowledge dissemination in autonomic networks", Communications (ICC), 2012 IEEE International Conference on, 10-15 juin 2012, Ottawa, Canada.

Tran H. A. ; Mellouk, A. ; Hoceini, S. ; **Souihi, S.**, "User QoE-based adaptive routing system for future Internet CDN", Computing, Communications and Applications Conference (ComComAp), 2012, 11-13 janvier 2012, Honk Kong, Chine.

Souihi, S. ; Mellouk, A., "Knowledge Dissemination for Autonomic Network", Communications (ICC), 2011 IEEE International Conference on, 5-9 juin 2011, Kyoto, Japon.

Conférences nationales avec comité de lecture et actes :

Souihi, S. ; Mellouk, A. ; Hoceini, S., “Dissémination des connaissances dans un réseau autonomique”, Journées Doctorales / Journées Nationales (JD-MAC) du GDR CNRS MACS, 6-8 juin 2011, Marseille, France.

Souihi, S. ; Mellouk, A., “Dissémination des connaissances dans un réseau autonomique”, colloque Automatique et Réseaux de Communication 2011 (ARC’11) du GDR CNRS MACS, 1er avril 2011, Paris, France.

Souihi, S. ; Mellouk, A., “Plan de connaissance pour les réseaux autonomiques”, École d’été du GDR CNRS ASR, 13-18 Juin 2010, Toulon-Hyères, France.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Glossaire

AE : *Autonomic Element*
AWS : *Amazon Web Service*
CDN : *Content Delivery Network*
CIM : *Common Information Model*
CLARA : *Clustering LARge Applications*
DDoS : *Distributed Denial of Service*
DoS : *Denial of Service*
DIs : Dispositifs intelligents
DHT : Table de hachage distribuée
DMTF : *Distributed Management Task Force*
EC2 : *Elastic compute Cloud*
IA : Intelligence Artificielle
ICN : *Information Centric Network*
IETF : *Institute of Electrical and Electronics Engineers*
INDL : *Infrastructure & Network Description Language*
JSON : *JavaScript Object Notation*
MAPE : *Monitor Analyse Plan and Execute*
MIB : *Management Information Base*
NExTLab : *Network Experimental Tools for research Laboratory*
NoSQL : *Not only SQL*
NTT : *Nippon Telephone and Telegraph Company*
PHT : Test statistique de Page-Hinkley
PAM : *Partitioning Around Medoïds*
PVE : *Proxmox Virtual Environment*
QdS : Qualité de Service
QdE : Qualité d'Expérience
RDF : *Resource Description Framework*
P2P : Pair-à-Pair
RTT : *Round Trip Time*

S3 : *Secure Storage Service*

SDN : *Software Defined Network*

SGBD : Système de Gestion de Base de Données

SMA : Système Multi-Agents

SPSA : *Super-Peer Selection Algorithm*

SPSP : *Super-Peer Selection Problem*

SSTable : *Sorted String Table*

Introduction

Contexte général de la thèse

L'architecture des réseaux informatiques actuels a été conçue pour répondre à un besoin simple : faire communiquer deux noeuds du réseau indépendamment du type de trafic ou de la nature des noeuds (simples machines ou humains). En effet, le réseau actuel est constitué de deux entités distinctes [Saltzer et al., 1991]. La première forme le coeur du réseau. Elle est en charge des couches basses et ne traite que des trames et des paquets. Quant à la seconde, elle est constituée des équipements de bordure qui se chargent de tous les processus de gestion et de contrôle. Cette conception, de par sa simplicité, a contribué à la croissance rapide du réseau Internet car elle a permis à toute machine de pouvoir se connecter au réseau et de communiquer avec toute autre machine.

Toutefois, cette même architecture semble aujourd'hui freiner l'évolution du réseau. Parallèlement à l'augmentation du nombre d'usagers d'Internet et à la diversification des terminaux et des moyens d'accès au réseau (voir figure 1), nous avons assisté à un changement des usages du réseau. En effet, alors qu'au début d'Internet, le type de données échangées sur le réseau était du simple texte, il est devenu aujourd'hui majoritairement multimédia et très demandeur en ressources. Paradoxalement, force est de constater aussi que les couches basses n'ont que très peu évolué.

Devant un tel constat, la plupart des opérateurs ont choisi de surdimensionner leurs réseaux repoussant ainsi les problèmes, notamment ceux liés à la congestion, aux frontières de leurs réseaux. Toutefois, cette approche n'est pas suffisante. En effet, le trafic est intrinsèquement très variable et se caractérise par des anomalies ayant pour conséquences des changements d'états imprévisibles dans le coeur du réseau. Par ailleurs, les profils usagers sont variés et les besoins de ces derniers sont multiples et extrêmement volatils.

La question qui se pose est donc la suivante : comment faire pour satisfaire au mieux l'usager ? Compte tenu des outils standards mis à disposition des administrateurs, il apparaît qu'il n'est plus possible de garder un temps de réaction acceptable face aux divers problèmes que subit le réseau. Ils sont, en effet, en grande majorité à configuration manuelle, les rendant ainsi inappropriés à un usage à large échelle et dans des situations où le temps de réaction

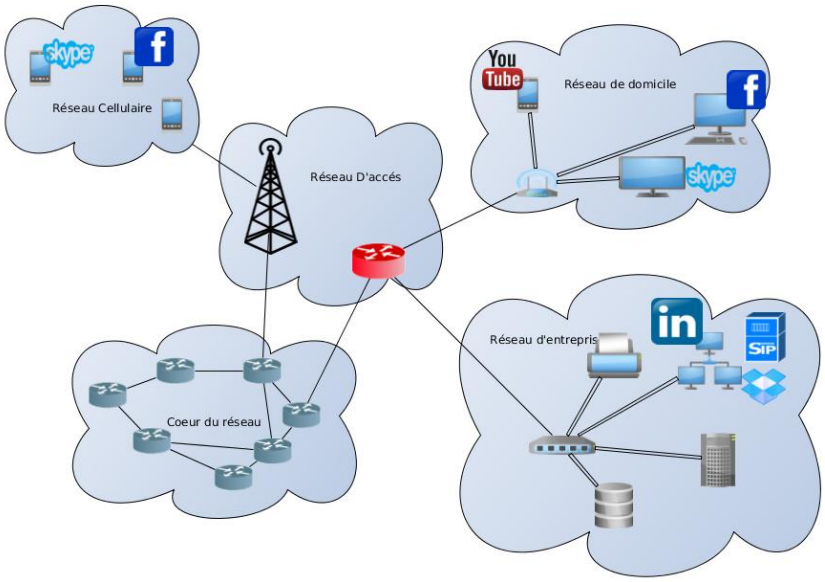


FIGURE 1 – Architecture du Réseau

est primordial.

Dans cette course vers l'amélioration de la qualité fournie aux usagers, nous pensons qu'on ne peut pas garder le coeur du réseau loin de toute complexité encore longtemps. Un changement fondamental de la conception du réseau doit avoir lieu : d'une vue "centrée réseau", nous devons évoluer vers une vue "centrée connaissance". Cette architecture doit pouvoir s'adapter à la nature fortement variable des paramètres d'un réseau ainsi qu'à la dynamicité des ressources disponibles. Une telle architecture est appelée parfois "architecture intelligente", "architecture adaptative", ou encore "architecture autonome". C'est ainsi que les approches développées ces dernières années ne se contentent pas seulement de manipuler des informations sur les plans de données, de contrôle ou de gestion, elles intègrent de plus en plus des connaissances, acquises ou apprises, sur différents paramètres définissant l'état du réseau tels que : le trafic, les ressources, les besoins, etc. Pour être efficace, de telles informations doivent être suffisantes et pertinentes et en même temps refléter de manière fiable l'état du réseau. Ces connaissances sont structurées dans ce qui est communément appelé "un plan de connaissance".

Celui-ci est un concept introduit par Clark et al. [2003]. Pour en construire un, disposant de caractéristiques robustes et fiables, nous devons répondre à plusieurs verrous tels que :

i) La définition de la connaissance,

ii) La proposition d'une représentation unifiée de la connaissance,

iii) L'énumération des moyens possibles pour la collecte des connaissances existantes et l'apprentissage de nouvelles connaissances,

iv) Et plus particulièrement, la proposition d'un moyen de distribution de cette connaissance à travers les noeuds du réseau en prenant en considération des contraintes comme le passage à l'échelle ou la robustesse du système.

Objectifs de la thèse

Nous proposons, à travers ces travaux de thèse, une plateforme de gestion des connaissances, générique et sous license libre, qui tient compte de critères liés à la fois à un déploiement à grande échelle et à l'interopérabilité. Elle s'intéresse particulièrement aux aspects de collecte, de représentation et de dissémination des connaissances dans un réseau.
Notre approche se base sur deux idées clés :

1. Une architecture hiérarchique et adaptative où un certain nombre de noeuds élus du réseau sont en charge de la gestion et du partage des connaissances.
2. La construction adaptative et dynamique de réseaux recouvrants (overlay) reliant les noeuds élus selon différents types de critères. Pour chacun de ces réseaux, un plan de connaissance spécialisé est instancié. Ce dernier peut correspondre à une application donnée ou encore à un besoin de prioriser certaines connaissances.

Cette plateforme générique, facilement déployable et reposant sur l'utilisation des réseaux recouvrants se veut comme une solution utilisable dans différents contextes. Mis à part la gestion des réseaux autonomes et le contrôle des réseaux logiciels, plus connus sous l'appellation anglo-saxone *Software Defined Networks* (SDN), notre plateforme peut être adaptée à un usage multiple et dans différents contextes d'utilisation, comme par exemple : les réseaux de diffusion de contenu (*Content Delivery Network*, CDN), les réseaux centrés sur l'information (*Information Centric Networks*, ICN), le Cloud Computing ou encore, les SmartGrids. À des fins de validation, nous avons sélectionné deux cas d'études : le premier concerne le domaine du Cloud Computing et le second traite des SmartGrids.

Nous nous sommes intéressés dans cette mise en oeuvre à l'optimisation d'un certain nombre de critères auxquels s'intéresse, traditionnellement, ce type de plateformes : la latence d'accès aux connaissances, le temps de dissémination des connaissances à tous les noeuds du réseau, le partage de charge et la surcharge générée (*overhead*).

Organisation de la thèse

Le présent document de thèse est organisé en 4 chapitres :

Chapitre 1 Ce chapitre introduit dans un premier temps le concept lié à l'autonomique dans les réseaux. Ensuite, nous discutons l'importance de la connaissance dans les réseaux autonomes et nous détaillons le fonctionnement du plan de connaissance. Ce chapitre se termine par l'énoncé de la problématique de la dissémination des connaissances et les différentes approches utilisées.

Chapitre 2 Ce chapitre présente l'approche développée dans le cadre de cette thèse pour la dissémination des connaissances dans le réseau. Cette dernière repose sur une architecture à base de super noeuds. Nous proposons de reconceptualiser le problème comme un problème de partitionnement K-medoïd et nous présentons les approches proposées.

Chapitre 3 Ce chapitre présente dans sa première partie la conception de la plateforme mise en place. La seconde partie est dédiée à la résolution du problème de construction d'une architecture à base de super noeuds. La troisième partie synthétise les résultats de simulation obtenus comparativement à d'autres approches existantes de gestion des connaissances. Deux cas d'usage de la plateforme dans le cadre des réseaux SDN sont ensuite traités. Nous concluons ce chapitre par une étude expérimentale portant sur le décomposition du plan global de connaissance.

Chapitre 4 Pour montrer la généricité de la plateforme mise en oeuvre, la première partie de ce chapitre décrit un cas d'usage relatif au domaine lié au Cloud Computing. La seconde partie est dédiée à la gestion des connaissances dans un système d'information correspondant aux Micro-Grids.

Ce mémoire se conclut par un ensemble de perspectives pour des recherches futures.

Chapitre 1

L'autonomique dans les réseaux et le besoin en connaissances

Sommaire

1.1 Introduction

Toute architecture doit pouvoir évoluer pour s'accommoder avec les nouvelles tendances, or, Internet n'a que très peu évolué depuis sa création. En effet, le réseau Internet a été conçu pour le transport d'un trafic de type texte très peu volumineux, alors qu'aujourd'hui, il est plutôt utilisé pour le trafic multimédia. Selon Cisco [2013], le trafic vidéo sur Internet représentera 69% de tout le trafic grand public en 2017. En 2012, il n'était que de 57%. Ce pourcentage sera de l'ordre de 80% à 90% du trafic mondial consommé d'ici 2017 en incluant les vidéos échangées via les réseaux pair-à-pair (P2P). Ce

type de trafic impose que l'acheminement des flux soit assuré avec une Qualité de Service (QdS) maîtrisée, chose que l'architecture actuelle des réseaux a du mal à fournir puisqu'elle n'a pas été conçue pour cela.

Partant de ce constat, on assiste, depuis peu, à un engouement de la communauté dans un effort d'améliorer l'architecture d'Internet de manière à l'adapter aux nouveaux types de contenus. Comme le suggèrent Ghodsi et al. [2011], plusieurs études ont été menées sur des réseaux centrés sur le contenu ou les données. Nous trouvons donc dans la littérature des appellations comme *Data-Oriented Network, Concent Centric Network, Content-based Network, Named-Data Network,...*. Malgré les divergences que peuvent avoir ces différentes propositions, elles peuvent être considérées comme faisant partie intégrante des réseaux centrés sur l'information (*Information Centric Networks*, ICN) dans la mesure où elles s'accordent sur deux points clés :

Le premier concerne un mécanisme de gestion du contenu cache. Celui-ci a deux comportements :

i) Si un noeud demande une donnée à un autre noeud qui l'a dans son cache, ce dernier lui renvoie directement la donnée en question,
ii) Si la donnée n'est pas en cache, ce dernier la demande à son tour et la met en cache.

Cette approche a des conséquences sur les mécanismes de sécurité. En effet, là où il suffisait de sécuriser le serveur original des données et le chemin emprunté par les paquets, il faut sécuriser dorénavant le contenu lui même dans une approche à base d'ICN. Cette sécurité est assurée par un mécanisme de certificats : le serveur original signe le contenu, ainsi tous les noeuds du réseau et les usagers peuvent, facilement, en vérifier la validité.

Le second point a trait au mécanisme de diffusion/récupération de l'information (ou requête/réponse). Certes, ce n'est pas le paradigme le plus récent étant donné qu'il est largement étudié depuis une vingtaine d'années, mais il s'agit d'un composant majeur des ICNs. Même si le nom de ce concept change d'une approche à une autre, celles-ci s'accordent toutes sur le fait que ce mécanisme doit être composé de deux primitives :

i) La publication des informations : Elle permet à un noeud de mettre à disposition de l'ensemble du réseau les informations à sa disposition.
ii) La récupération des informations : Elle consiste à permettre à tout noeud de faire des requêtes sur les informations publiées.

Toutefois, et afin de réponde rapidement et de la manière la plus adéquate aux besoins changeants et volatils des utilisateurs, ces nouvelles approches augmentent la complexité du contrôle dans le réseau. Le nouveau paradigme des réseaux logiciels, plus connus sous l'appellation anglo-saxone *Software Defined Networks* (SDN), tente de répondre à cette problématique [Open-Networking-Foundation, 2012]. Il s'agit d'une nouvelle approche dans laquelle le contrôle est découplé du matériel et donné à un logiciel appelé contrôleur. Le but des SDNs est de permettre aux ingénieurs et aux administrateurs de construire des "réseaux à la carte". Les réseaux, ainsi construits, sont pilotés à partir d'un pupitre de commande "centralisé" sans avoir à intervenir sur les commutateurs individuellement. Toutefois, la nature des flux circulant sur le réseau et la constante progression des besoins des utilisateurs font qu'un contrôle centralisé et en partie manuel n'est pas viable à long terme. Il nous semble donc nécessaire d'opérer une évolution technologique du réseau. Cette évolution, qui consiste à intégrer une automatisation de certaines tâches dans la gestion du réseau, est appelée approche autonomique.

1.2 Approche autonomique

Les réseaux autonomiques représentent un concept qui a pour ambition de rendre le réseau indépendant de tout pilotage humain. Une telle approche répond parfaitement à la problématique actuelle qui, compte tenu de l'accroissement des besoins, consiste à continuer à offrir aux utilisateurs un service fiable et sans couture, connu sous l'appellation anglo-saxone "seamless service" (le système doit gérer dynamiquement la session de l'utilisateur et son unicité de bout en bout en temps réel). L'idée repose sur une adaptation du concept du système autonome introduit par IBM en 2001 [Kephart and Chess, 2003 ; Kephart, 2005] à la thématique des réseaux informatiques. Ce type d'approche a pour ambition de répondre à la complexité grandissante des réseaux et de permettre ainsi leur expansion future au-delà de leur taille actuelle.

1.3 Définition des réseaux autonomiques

Les réseaux autonomiques sont des réseaux qui offrent des capacités "d'auto-*" sans aucune intervention externe. D'un point de vue architectural, ils sont basés sur un ensemble d'éléments autonomes appelés *Autonomic Elements* (AEs). Comme décrit dans la figure 1.1, un AE a 4 caractéristiques

principales [Sterritt et al., 2005] :

— se connaître soi-même (*self-awareness*),
— connaître son environnement (*environment-awareness*),
— s'auto-monitorer (*self-monitoring*),
— s'auto-ajuster (*self-adjusting*).

Ces caractéristiques constituent les conditions nécessaires pour qu'un système autonome puisse assurer les 4 objectifs décrits par Sterritt et al. [2005] :

— auto-configuration (*self-configuring*),
— auto-réparation (*self-healing*),
— auto-optimisation (*self-optimising*),
— auto-protection (*self-protecting*),

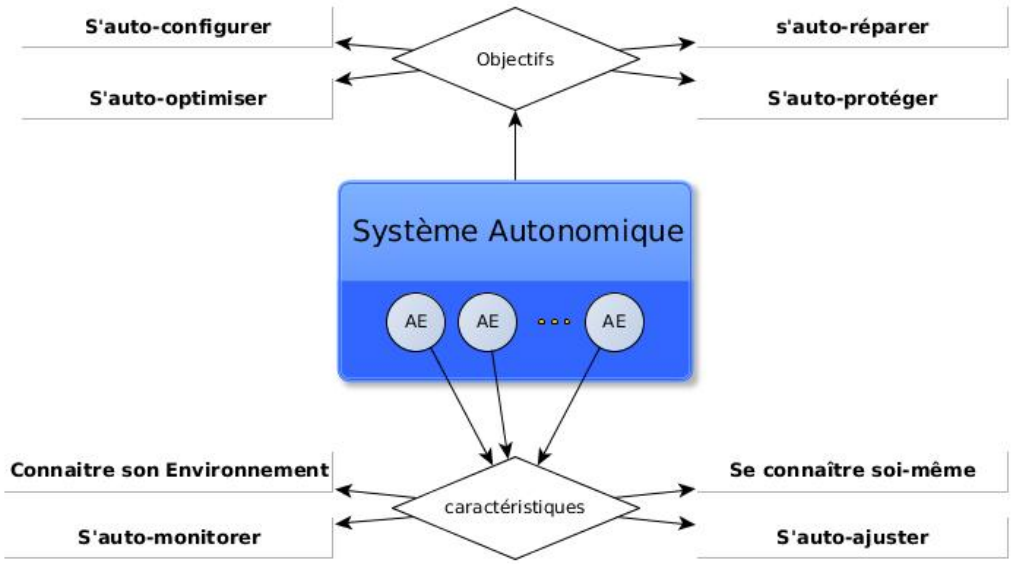


FIGURE 1.1 – Propriétés d'un système autonomique

1.4 Caractéristiques d'un réseau autonomique

Du point de vue architectural, un AE se doit de répondre à quatre caractéristiques : *self-awareness*, *environment-awareness*, *self-monitoring* et *self-adjusting* [Kephart and Chess, 2003 ; Kephart, 2005]. Nous définissons, dans la suite, chacune d'entre elles :

- ***Self-awareness*** est la capacité d'un individu de se connaître soi-même indépendamment de l'environnement et des autres individus.

  D'un point de vue philosophique, la connaissance de soi même peut être résumée par la citation : « cogito ergo sum » (Je pense, donc je suis).

  Du point de vue purement technique, un AE doit donc avoir connaissance de ses propres ressources, ses composants, ses performances et son état interne.
- ***Environment-awareness*** (connaître son environnement) consiste à connaître l'ensemble des mécanismes avec lesquels le système est en interaction. Cette caractéristique a une incidence sur la faculté du système à s'adapter aux différents contextes.
- ***Self-monitoring*** (auto-monitoring) est la faculté d'un système de suivre par soi-même et régulièrement son comportement et son évolution en vue de pouvoir évoluer et s'améliorer en fonction du contexte.
- ***Self-adjusting*** (auto-ajustement) est la capacité d'un système de s'adapter au changement de son environnement et de rétablir ses fonctionnalités après perturbation, sans aucune intervention externe.

1.5 Objectifs d'un réseau autonomique

Du point de vue fonctionnel, un système autonome se doit d'assurer quatre automatismes à la fois : l'auto-réparation, l'auto-configuration, l'auto-optimisation et l'auto-protection [Kephart and Chess, 2003 ; Kephart, 2005]. Nous définissons, dans la suite, chacune d'entre elles :

- **Auto-réparation :** Les systèmes qui s'auto-réparent peuvent automatiquement identifier, analyser, résoudre et réparer les problèmes. L'objectif étant de maintenir de façon continue la disponibilité des réseaux et des services et de gagner du temps par rapport aux approches actuelles où parfois un certain temps est requis pour le traitement du problème. La plateforme *Open Object Request Broker* (OpenORB) [Blair et al., 2002] décompose le mécanisme d'auto-réparation en deux entités :
  — La première, dite entité de monitoring, observe le comportement des couches fonctionnelles sous-jacentes, collecte les informations liées à la Qualité de Service (QdS) et fait part d'un comportement anormal,
  — La deuxième, dite entité de contrôle, s'occupe de sélectionner une stratégie d'adaptation appropriée basée sur les rapports émis par le mécanisme de monitoring.

- **Auto-configuration :** Elle couvre le besoin de rendre la configuration et la reconfiguration d'un système et de ses entités dynamiques et autonomes, en particulier, dans les phases d'installation et de suppression d'un nouveau composant dans le système. Ce dernier pourra ainsi auto-déterminer la bonne configuration à adopter et éviter les nombreuses erreurs de configuration liées au facteur humain.

  L'auto-configuration doit être réalisée dans le but d'atteindre le comportement attendu. Citons comme exemple la proposition de Molinier et al. [2012] concernant l'auto-configuration des équipements dans un réseau de domicile. Dans cette approche, un agent intelligent est placé sur chaque équipement. Il est censé s'occuper de la configuration des interfaces, de la découverte du voisinage et du choix du chemin et du médium de communication d'une manière tout à fait transparente à l'utilisateur.

- **Auto-optimisation :** L'optimisation d'un système est une tâche difficile à effectuer compte tenu du nombre de paramètres en jeu. Afin d'éviter à l'administrateur de paramétrer lui même les équipements et les réseaux pour fonctionner correctement, les systèmes doivent continuellement chercher à améliorer leurs performances et optimiser leurs opérations. L'auto-optimisation concerne donc l'automatisation de la gestion des performances des systèmes. Celles-ci doivent être optimisées et évaluées de façon continue.

- **Auto-protection :** Elle concerne les capacités d'un système à se protéger et à se sécuriser. L'automatisation de la sécurité a pour but de développer un comportement pouvant détecter les situations où la stabilité des réseaux et des services peut être remise en cause par le biais d'actions volontaires ou non et de se protéger en conséquence pour ne pas perturber les usagers.

  Comme exemple, Luo et al. [2002] proposent un service distribué d'authentification dans le contexte des réseaux ad-hoc. Dans cette solution, de multiples nœuds collaborent afin de traduire le comportement d'un serveur fournissant la certification et l'authentification pour d'autres nœuds du réseau ad-hoc.

- **Autres fonctions** D'autres auto-fonctions ont été définies dans divers travaux de recherche (auto-gestion, auto-localisation, ...) au regard d'un besoin particulier du contexte étudié. Généralement, cette diversification mène à un chevauchement des objectifs. Par exemple, l'auto-gestion couvre à la fois certains attributs de l'auto-configuration et d'autres relatifs à de l'auto-optimisation.

  Après avoir présenté les caractéristiques et les objectifs d'un système autonome, nous décrirons, dans la suite, l'architecture de ce dernier.

1.6 Architecture d'un réseau autonomique

Un système autonome est constitué d'une collection d'éléments autonomes (*Autonomic Elements*, AEs), capables de gérer leur comportement interne ainsi que les relations avec les autres éléments autonomes sans aucune intervention humaine. Pour ce faire, ils s'appuyent sur la connaissance acquise ou apprise par le biais des approches déductives ou inductives déployées dans le réseau. En Intelligence Artificielle (IA), le module qui se charge d'une telle gestion est nommé "agent".

1.6.1 Vision agent du paradigme

Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs pour lesquels il a été conçu [Wooldridge and Jennings, 1994].

En distribuant les agents sur chaque équipement (comme le montre la figure 1.2), le système peut alors traiter localement tout problème l'impactant localement et du coup, diminuant de fait sa répercussion sur le système global. Dans la majorité des situations, les problèmes sont plus facilement gérables quand ils sont contenus localement. De plus, un problème local est détecté beaucoup plus rapidement que dans une approche dite "centralisée".

Pour accomplir sa tâche, un agent doit être [Wooldridge and Jennings, 1994] :

- **Situé :** l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement,
- **Autonome :** l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne,
- **Proactif :** l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment,
- **Réactif :** l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis,
- **Social :** l'agent doit être capable d'interagir avec d'autres agents (automatiques ou humains).

Un ensemble d'agents interconnectés entre eux forment un Système Multi-Agents (SMA). Ce système peut doter la machine de l'intelligence requise et des capacités d'apprentissage lui permettant de tirer profit d'une expérience particulière pour faire face à une nouvelle situation.

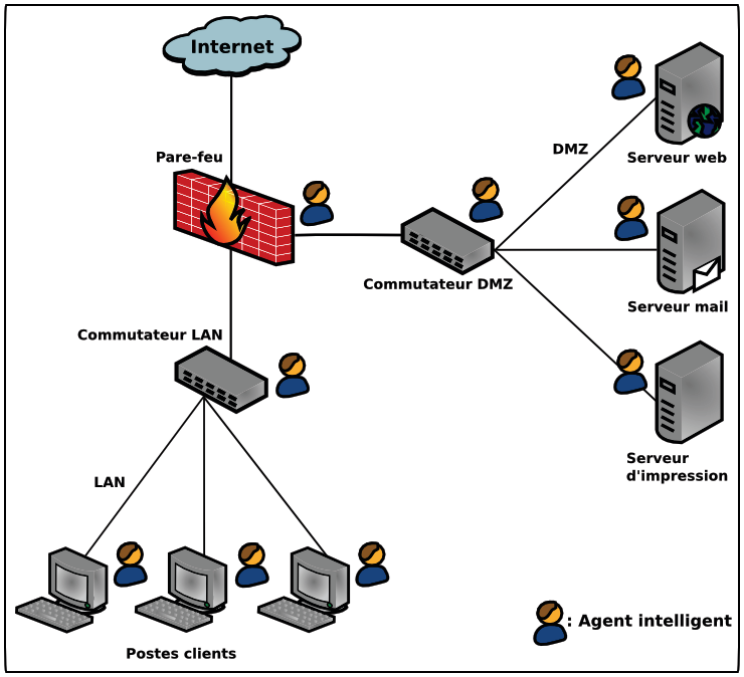


FIGURE 1.2 – Déploiement des agents dans un réseau

Les caractéristiques principales d'un SMA sont :

— Chaque agent a des informations ou des capacités de résolution de problèmes limitées,
— Il n'y a aucun contrôle global du système multi-agents,
— Les données sont décentralisées,
— Le calcul est asynchrone,

Un SMA fournit donc l'intelligence nécessaire pour comprendre le comportement du réseau afin qu'il puisse s'auto-piloter. Il est néanmoins nécessaire de lui fournir un ensemble de connaissances pour offrir des capacités d'auto-gestion élevées et efficaces. Ces connaissances doivent être apprises et acquises à partir de l'ensemble des agents et partagées entre eux de la manière la plus optimisée possible.

1.6.2 Architecture d'un élément autonome

Selon [Kephart and Chess, 2003 ; Kephart, 2005], un élément autonome (*Autonomic Element*, AE) est constitué de deux composants : une ressource gérée et un agent autonome. L'agent autonome est constitué d'une boucle de contrôle (*Control loop*) appelé MAPE (**M**onitor, **A**nalyse, **P**lan and **E**xecute)

capable de gérer le comportement de la ressource. L'interaction entre les deux composants de l'AE se fait à travers une interface constituée de capteurs (Sensors) et d'actionneurs (Effectors). Une interface identique sert à l'interaction de l'AE avec l'environnement extérieur. Les deux interfaces, la MAPE et la ressource sont reliées par un ensemble de connaissances qui décrivent l'état actuel et passé du AE et de son environnement.

La figure 1.3 représente l'architecture d'un AE. Dans ce qui suit, nous définissons chacun de ses composants.

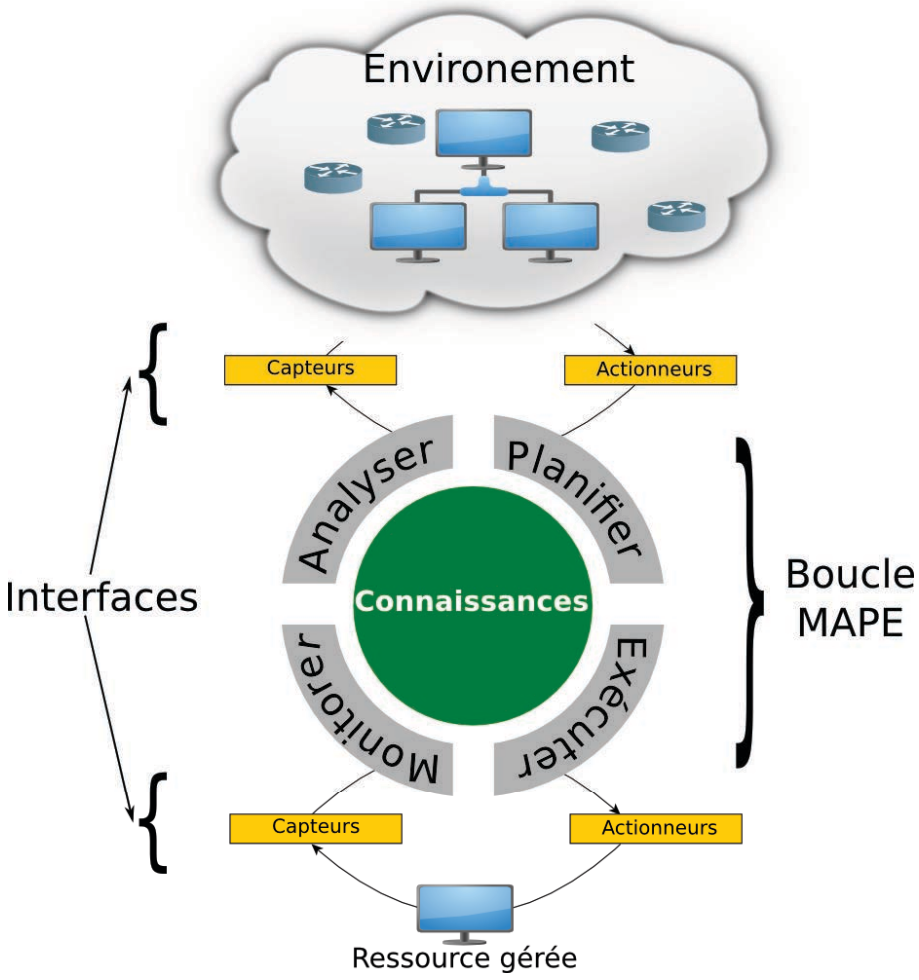


FIGURE 1.3 – Architecture d'un système autonome

1.6.2.1 Ressource

Une ressource gérée est définie comme tout équipement (logiciel ou matériel) pouvant être contrôlé à travers des actionneurs et supervisé à travers des capteurs. Une ressource peut donc être un noeud réseau (routeur, hub, commutateur, ...), un composant (RAM, disque, CPU, ...) ou encore une application (serveur de BD, application de vidéo à la demande, ...).

Toutefois, une ressource doit pouvoir être pilotée au travers d'actionneurs et pouvoir être supervisée au travers de capteurs.

1.6.2.2 L'interface

L'interface est ce qui permet, à la fois, l'interaction entre l'agent et la ressource gérée, et entre l'AE et l'environnement extérieur. Elle est structurée en un ensemble de capteurs et d'actionneurs. L'implémentation de chacun d'entre eux est spécifique à la ressource. L'intérêt de l'interface est donc de standardiser la gestion des ressources et d'améliorer l'interopérabilité.

Capteur : Il s'agit d'un mécanisme pouvant collecter des informations sur un élément géré. Dans les réseaux informatiques, nous pouvons distinguer deux types de capteurs :

- **Les Capteurs actifs** dont le fonctionnement consiste à générer du trafic dans le réseau et à observer l'effet produit sur ses différents composants. Ces capteurs sont intrusifs mais permettent généralement d'obtenir des mesures précises.
- **Les Capteurs passifs** dont le fonctionnement consiste à observer le trafic et les différents composants du réseau afin d'en déduire une certaine compréhension. Ces capteurs sont non-intrusifs, mais les mesures qu'ils fournissent sont, dans la plupart des cas, moins bonnes que celles fournies par leurs homologues actifs.

Actionneur : Il s'agit d'un mécanisme permettant d'agir sur l'élément géré. En d'autres termes, il offre les moyens nécessaires à l'exécution de tâches, éventuellement programmées, d'un système automatisé.

1.6.2.3 La boucle MAPE

La boucle MAPE (**M**onitor, **A**nalyse, **P**lan and **E**xecute) est le composant qui régit le comportement d'un AE. Les parties "Monitorer" et "Analyser" fournissent les caractéristiques liées aux fonctionnalités self-awareness et environnement-awarness. Les deux autres parties "Planifier" et "Executer" décident et réalisent la tâche de self-management (grâce aux actionneurs).

Le Monitoring : Il consiste en un ensemble de mécanismes pouvant collecter des informations sur un équipement géré. Ces informations peuvent être des paramètres définissant l'état du réseau (trafic, ressources, besoins, ...), les caractéristiques du réseau (topologie, capacité) ou ceux liées à l'utilisateur (périphérique, besoins, ...).

La collecte de ces informations se fait au travers de capteurs. Ces informations peuvent être filtrées, analysées ou bien agrégées avant d'être transmises au module d'analyse.

L'analyse : Elle consiste à observer et à évaluer le fonctionnement d'un système en vue de décider si une action doit être entreprise pour garantir un objectif de haut niveau.

La planification : Cette opération consiste à organiser les actions à entreprendre selon une méthodologie précise dans le but d'atteindre un objectif. Il s'agit de l'étape de formalisation des changements envisagés durant la phase d'analyse.

L'Exécution : Elle consiste à réaliser les actions décidées lors de la phase de planification. Cette réalisation se fait par le biais d'actionneurs. Le résultat obtenu permet d'enrichir les connaissances du système.

1.6.3 Systèmes autonomes existants

Plusieurs travaux de recherche ont été menés dans le but de produire des plateformes autonomes capables de gérer les ressources d'un réseau informatique. Nous en citerons dans la suite quelques exemples.

— **BIONETS :** Le projet BIONETS (**BIO**logically-inspired autonomic **NET**work**S**) [Carreras et al., 2007] s'inspire des mécanismes du monde biologique afin de mettre en oeuvre un système capable de gérer un grand nombre d'éléments hétérogènes sans aucune intervention humaine. Ce système dote les éléments du réseau d'une capacité d'évolution bio-inspirée en leur fournissant la faculté d'auto-adaptation due à l'interaction et à la coopération entre ses différents éléments.

— **GANA :** L'architecture GANA (**G**eneral **A**utonomic **N**etwork **A**rchitecture) a été proposée dans le cadre du projet EFIPSAN [Chaparadza et al., 2009 ; A. Liakopoulos, 2009]. Il s'agit d'une architecture qui implémente des fonctions d'auto-monitoring et d'auto-gestion. GANA est conçue selon une architecture cubique qui comprend quatre plans fonctionnels (plan de données, plan de découverte, plan de décision et plan de dissémination de connaissance). Chaque plan contient quatre niveaux d'abstraction (niveau réseau, niveau noeud, niveau fonction et niveau protocole) (cf. figure 1.4).

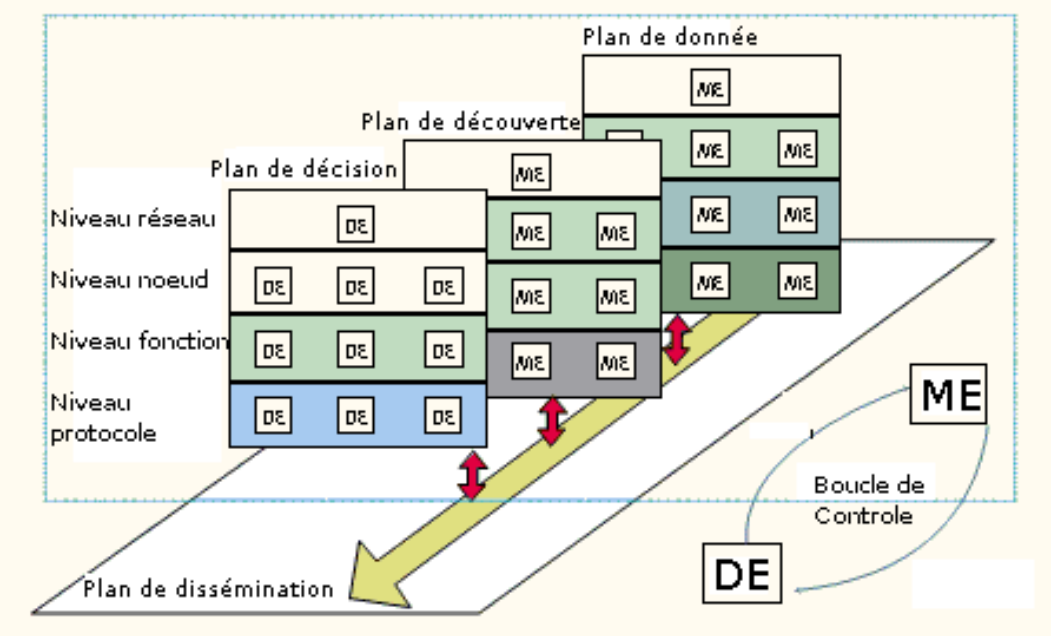


FIGURE 1.4 – Architecture de GANA [A. Liakopoulos, 2009]

GANA utilise les réseaux pair-à-pair dans le but de contrôler les performances et de gérer les ressources d'un système. Ce contrôle est effectué par des agents, appelés *Decision Element* (DEs), qui décident du comportement à adopter. Pour y parvenir, ils utilisent un ensemble d'informations stockées et disséminées à travers une table de hachage

distribuée (DHT) sur l'ensemble des noeuds du réseau.

— **FOCALE :** Strassner et al. [2006] ont proposé l'architecture FOCALE (**F**oundation, **O**bservation, **C**omparison, **A**ction and **L**earning **E**nvironment). Elle est l'une des premières architectures à utiliser les ontologies pour la représentation des connaissances. Comme GANA, FOCALE se base sur une table de hachage distribuée pour gérer ces connaissances.

— **ANA :** Le projet ANA (**A**utonomic **N**etwork **A**rchitecure) [Bouabene et al., 2010] vise à identifier les principes fondamentaux régissant les réseaux autonomes. Dans le but de démontrer la faisabilité d'un réseau autonome, ANA propose une plateforme de démonstration qui implémente ces principes et qui permet un déploiement à large échelle. L'architecture d'ANA repose sur deux composants principaux :
  — Blocs (Bricks) : Ils représentent les composants les plus atomiques d'ANA. Chaque bloc est en charge d'une fonction particulière. Les blocs Ethernet et IP fournissent par exemple le moyen d'interagir avec les réseaux existants.
  — Minimex : Il s'agit de l'élément qui permet aux briques d'interagir. Il contient toutes les unités de gestion permettant à une brique de découvrir les autres blocs locales et d'échanger des messages et des instructions.

  La dissémination des connaissances dans ANA n'est pas clairement définie. Toutefois Schuetz et al. [2007] étudie la gestion décentralisée des réseaux sans fil et propose d'utiliser un modèle de synchronisation dans un voisinage (portée d'une cellule radio).

— **CASCADAS :** Le Projet CASCADAS (**C**omponentware for **A**utonomic **S**ituated-aware **C**ommunication and **D**ynamic **A**daptable **S**ervices) [Marrow and Manzalini, 2006 ; Baresi et al., 2009] propose une plateforme de développement de réseaux autonomes à base d'ACEs (Autonomic Communication Elements) (cf. figure 1.5) capable d'assurer plusieurs auto-fonctionnalités (auto-organisation, auto-configuration, auto-protection et auto-optimisation). Les réseaux ainsi développés devraient être capables d'évoluer dans un environnement dynamique.

— **AUTOI :** Le projet AUTOI (**AUTO**nomic **I**nternet) [Galis et al., 2009 ; Abid et al., 2008] propose une solution autonome basée sur les réseaux virtuels recouvrants (*overlay networks*) peuvant être déployée de manière transparente sur le réseau Internet. Ces réseaux recouvrants permettent non seulement d'interconnecter plusieurs réseaux hétérogènes et mais aussi de supporter la mobilité.
  L'architecture d'AUTOI se compose de 4 plans :

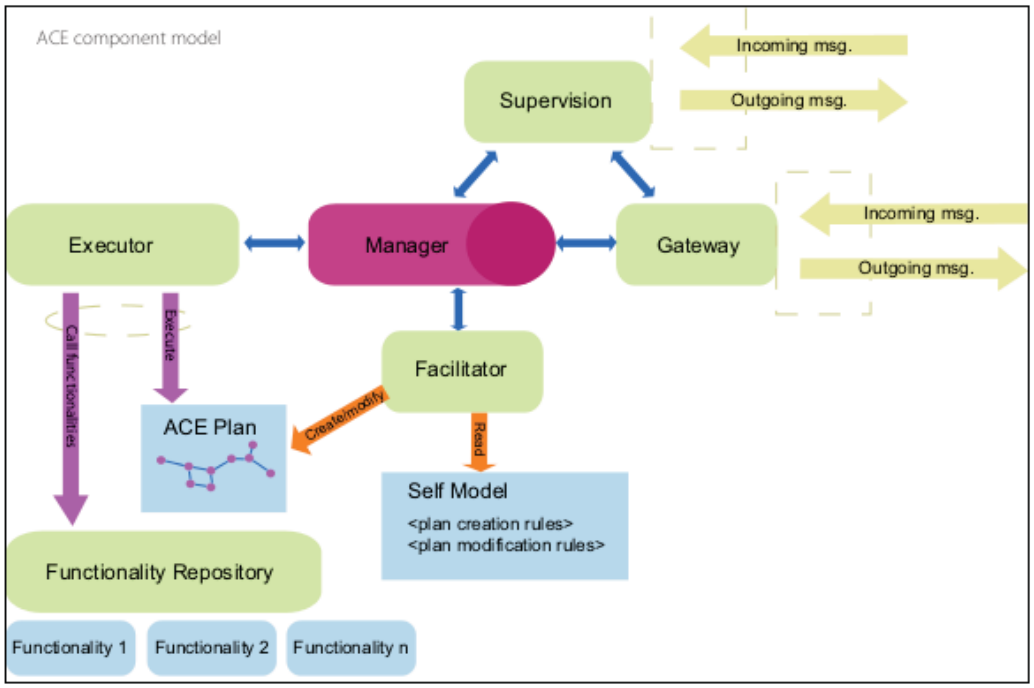


FIGURE 1.5 – Architecture de CASCADAS [Marrow and Manzalini, 2006]

— Plan de gestion : Il s'agit du plan en charge des différentes fonctions autonomes du système.
— Plan de connaissance : Il s'agit d'une base de données distribuée à travers l'ensemble des noeuds du réseau. Les données publiées y sont présentées sous la forme d'une ontologie commune. La distribution de la base de données se fait sur la base d'une approche correspondante à une "vue située" (cf. section 1.7.3.4).
— Plan d'orchestration : Il s'agit d'un plan de gestion capable de gérer plusieurs domaines et de résoudre, ainsi, les différents conflits qui peuvent subsister.
— Plan d'activation de service : Il fournit des fonctions pour le déploiement automatique ou l'activation de nouveaux services, de nouveaux protocoles, ainsi que des nouvelles ressources dans le réseau.

Chaque contribution dans les réseaux autonomes a apporté son lot de nouveautés : Utilisation des ontologies dans FOCALE [Strassner et al., 2006], Virtualisation dans AutoI [Abid et al., 2008] ou bien Conception atomique dans ANA [Bouabene et al., 2010]. Malgré les divergences que peuvent avoir ces différentes propositions, toutes s'accordent sur la nécessité d'une plateforme distribuée de gestion de connaissances. Cette plateforme constitue le

plan de connaissance. Il est donc intéressant de concevoir un nouveau système autonome ayant pour point de départ une plateforme de gestion de connaissances efficace et qui agrège une partie des innovations faites dans les autres propositions.

1.7 Plan de connaissance

Le plan de connaissance a été introduit par Clark et al. [2003] dans l'objectif de remédier aux limites conceptuelles de l'architecture IP classique. Son objectif a été de répondre à la question suivante : "Quelle serait l'architecture du réseau Internet si nous devons la refaire aujourd'hui ?". En fait, les deux objectifs majeurs du réseau Internet et qui ont fait son succès, la simplicité et la transparence, constituent aujourd'hui un frein à son évolution.

En plus clair, l'architecture des réseaux informatiques a été conçue pour répondre à un besoin simple : faire communiquer deux noeuds du réseau indépendamment du type de trafic qui transite et de la nature de ces noeuds. Dans cette architecture, le coeur du réseau se limite à retransmettre des paquets pour assurer leur transport [Saltzer et al., 1991]. Afin de ne pas encombrer les éléments de contrôle du réseau et, du coup, les temps de traitement, il a été décidé que toute la complexité et l'intelligence soient reléguées aux équipements de bordure du réseau. Ce principe répond essentiellement à deux objectifs majeurs :

— La transparence dans le fonctionnement du réseau : les noeuds qui communiquent n'ont pas à se soucier du fonctionnement des couches basses.
— La simplicité du déploiement : tout déploiement d'applications peut se faire sans aucune modification dans les couches sous-jacentes.

Ces attributs, certainement pertinents pour un réseau de taille contenu, comme le réseau Internet à ses débuts, se révèlent aujourd'hui comme des limites à son développement. En effet, la sécurité est remise en cause du fait des virus et autres logiciels malveillants qui usent de la simplicité du réseau pour se propager. De plus, la lenteur et l'inadaptation des prises de décision pour la gestion et l'optimisation du réseau rendent impossible tout type de contrôle réactif d'un réseau ayant subi des événements imprévisibles ayant impacté ses performances. Cela est dû au fait que ces décisions sont prises par un équipement de bord qui n'a aucune information pertinente sur le coeur du réseau et qui ne constate les anomalies que tardivement. Le rajout d'un plan

de connaissance à l'architecture réseau va permettre d'y remédier.

1.7.1 Architecture d'un plan de connaissance

L'architecture des réseaux est représentée, traditionnellement, sous la forme de trois "plans" (cf. figure 1.6) :

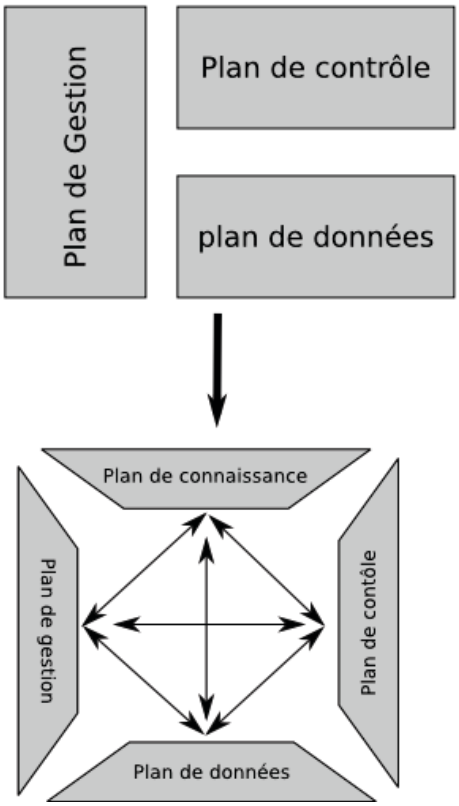


FIGURE 1.6 – Évolution de l'architecture réseau

Le plan de données : Il prend en charge le transit des données échangées entre les utilisateurs (exemples de fonctionnalités : ordonnancement, gestion de files d'attente, contrôle de congestion, façonnage de trafic, marquage de trafic, …).

Le plan de contrôle : Il permet de distribuer les politiques de contrôle et de gérer les données en temps réel (exemples de fonctionnalités : routage, signalisation, contrôle d'admission, réservation de ressources, …).

Un plan de gestion : C'est un plan transversal aux deux autres. Il est en charge de la supervision du plan de données et fournit une vue globale du bon fonctionnement du système (exemples de fonctionnalités : contrôle de supervision de QdS, configuration de QdS, politique de QdS, …).

Sur la base de cette architecture, chaque algorithme embarqué dans les noeuds du réseau doit donc s'adapter aux politiques demandées en fonction du contexte disponible à son niveau. Partant du constat que la majorité des informations nécessaires aux algorithmes de gestion du réseau est redondante, il s'agit donc de les agréger et de les porter à la connaissance des noeuds qui en font la demande explicite. Notons ici que la périodicité avec laquelle ils nécessitent ces informations n'est pas forcément la même. L'ajout d'un plan de connaissance à l'architecture réseau rend cet aspect possible. De plus, il devient aisé de définir des objectifs globaux qui peuvent s'adapter aux différents contextes locaux. À haut niveau, le plan de connaissance est chargé d'agréger les observations et les contraintes afin d'y appliquer un raisonnement et de générer des réponses pour les diverses situations auxquelles peut être confronté un réseau informatique. Ce nouveau plan requiert la coopération d'un ensemble de noeuds du réseau dans le but de partager les informations nécessaires à leur bon fonctionnement ainsi que les connaissances acquises par chacun d'entre eux. Chaque élément du réseau peut alors choisir la politique à appliquer en fonction de son contexte.

L'architecture d'un plan de connaissance réside dans le concept de séparation des couches [Clark et al., 2003]. En effet, le plan de connaissance n'a pas pour objectif de se substituer aux autres plans, mais confère aux mécanismes de contrôle les moyens de répondre à des objectifs globaux. Il s'agit ici d'un changement total de l'architecture réseau qui passe d'une vue "stratifiée" à une vue "*cross-layer*" comme le montre la figure 1.6. Cette approche permet de faire évoluer la logique de la gestion des réseaux en rendant la complexité locale transparente et en ajoutant un nouveau niveau d'abstraction au-dessus des règles et des politiques de contrôle.

1.7.2 Les concepts clés d'un plan de connaissance

D'après Clark et al. [2003], il faudra reconstruire le réseau autour d'un plan de connaissance afin que celui-ci soit le plus efficace possible. À défaut de pouvoir le faire, nous devons identifier les outils disponibles pour la mise en place de ce plan ainsi que les attributs clés auxquels il doit répondre. Ces attributs sont représentés dans la figure 1.7. Nous définissons, dans la suite,

chacun d'entre eux.

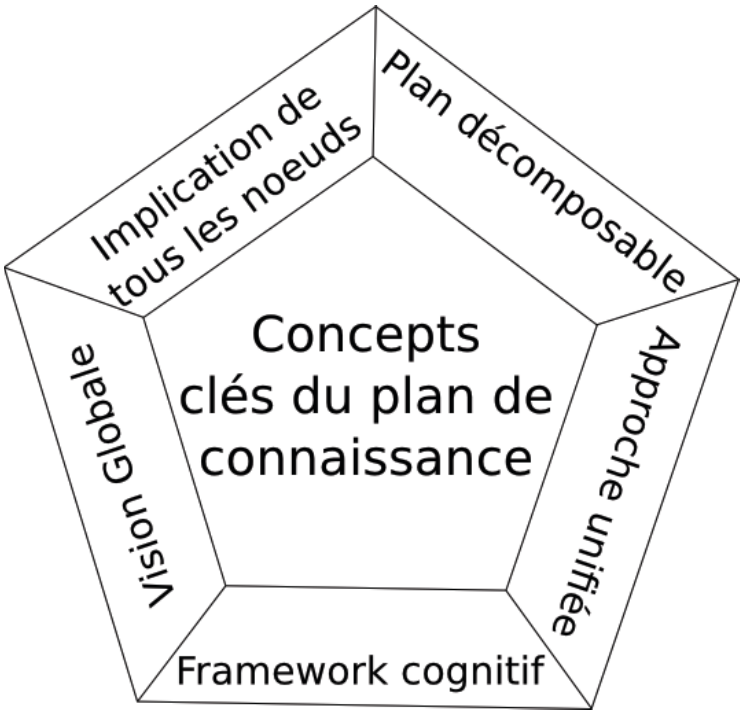


FIGURE 1.7 – Concepts clés d'un plan de connaissance

1.7.2.1 Système cognitif

Les techniques cognitives peuvent servir de fondation à la mise en oeuvre d'un plan de connaissance. En effet, celui-ci a besoin de porter des jugements en présence d'informations partielles ou contradictoires afin de :

- reconnaître et arbitrer les conflits dans les politiques et les objectifs,
- répondre aux dysfonctionnements ou à tout événement impactant le réseau dans des délais très courts,
- effectuer des optimisations d'environnements de grande dimension, dont le traitement, par des administrateurs ou des solutions analytiques classiques, demeurent compliqués,
- automatiser les fonctions qui nécessitent des ressources humaines hautement qualifiées.

Selon Brachman [2002], un système cognitif est un système qui, *"en plus de pouvoir raisonner et apprendre à partir de l'expérience pour améliorer ses performances au cours du temps, peut expliquer et justifier les actions qu'il*

entreprend". Les fonctions d'un système cognitif selon Vernon et al. [2007] peuvent être résumées comme suit : la perception, l'action, l'anticipation, l'adaptation et la motivation.

1.7.2.2 Approche unifiée

Les mécanismes distincts et localisés sont faciles à mettre en oeuvre et la solution à court terme peut être efficace. Toutefois, une approche unifiée est plus efficace à moyen et long termes. En effet, la connaissance du monde réel ne peut pas être strictement découpée en tâches indépendantes et distinctes. Le plan de connaissance doit donc être conçu comme un système unifié.

1.7.2.3 Prise en compte de la connaissance au niveau des extrémités du réseau

Le principe de bout en bout impose que beaucoup d'informations pertinentes sur les performances du réseau ne proviennent pas du coeur du réseau, mais des périphériques et des applications qui l'utilisent. Une partie des connaissances est donc produite, gérée et consommée dans les extrémités du réseau. Elles sont totalement transparentes par rapport aux éléments situés dans le coeur du réseau. Si ce dernier détecte une anomalie, il doit se contenter de raisonner sur une connaissance partielle du problème. L'objectif du plan de connaissance est entre autres celui d'injecter une partie de la connaissance issue des équipements de bordure du réseau dans le contrôle des éléments situés au delà.

1.7.2.4 Vue globale

La plupart des systèmes de gestion sont décentralisés : chaque opérateur gère la partie dont il est propriétaire. Toutefois, l'identification de certains problèmes dépend de la corrélation de différentes observations. Non seulement les données provenant des éléments de bordure doivent être combinées avec les données du coeur du réseau, mais les données de différentes régions (représentées par des différents opérateurs) du réseau peuvent être nécessaires pour reconstruire le puzzle qui constitue la suite d'événements impactant le réseau. Bien qu'utopique, le plan de connaissance doit, dans l'idéal, être en mesure d'étendre sa perspective pour l'ensemble du réseau mondial.

1.7.2.5 Plan décomposable

Un plan de connaissance doit non seulement offrir une vue globale du réseau, mais permettre aussi de structurer cette connaissance en un ensemble de sous plans indépendants. Cette granularité répond à plusieurs objectifs :

La sécurité : Certaines données sensibles ne peuvent pas être divulguées par un opérateur. Il est donc nécessaire de garder la possibilité de gérer une connaissance privée.

Le passage à l'échelle : Certaines connaissances ne sont nécessaires que pour un raisonnement local et il n'est donc pas nécessaire de les partager avec le reste du réseau.

La cohérence : L'incohérence peut résulter de la corrélation de certaines connaissances issues de contextes différents. Pour éviter une possible mauvaise interprétation, certaines connaissances doivent par ailleurs être cloisonnées.

Toutefois, il n'en demeure pas moins que des sous-plans non connectés doivent aussi être capables de fusionner leur point de vue et leurs connaissances si nécessaire. C'est le principe d'interopérabilité.

1.7.3 Verrous technologiques du plan de connaissance

Afin de développer un plan de connaissance qui permet de répondre aux concepts clés décrits par Clark et al. [2003], nous devons d'abord relever certains défis et verrous technologiques que nous détaillons ci-dessous :

1.7.3.1 Définition de la connaissance

Au regard de l'usage de certains termes de plus en plus répandus dans les publications scientifiques, il n'est pas trivial de différencier les données, les informations et les connaissances. Afin de lever cette confusion, il est important de préciser que le cycle de vie d'une mesure ou d'une observation passe par 3 étapes comme le montre la figure 1.8.

Chaque étape fait appel à un concept précis décrit ci-dessous :

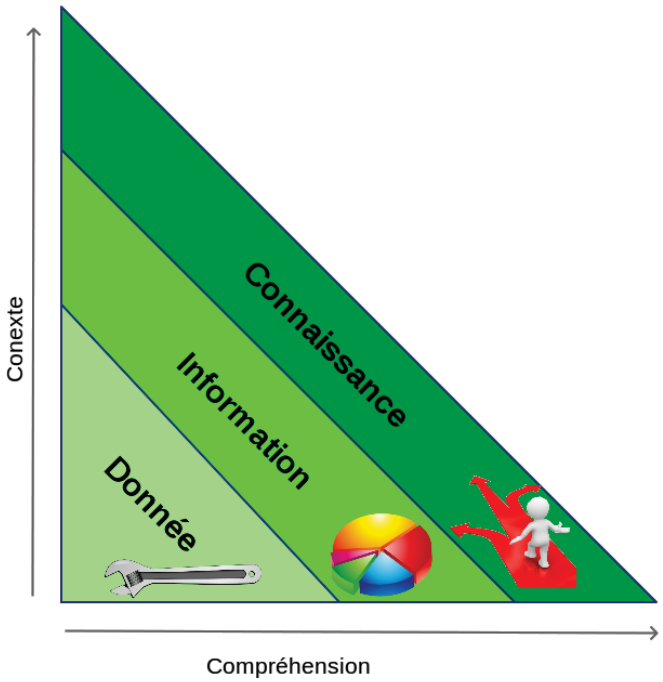


FIGURE 1.8 – Cycle de vie d'une donnée

1.7.3.1.1 Donnée Selon la définition du Larrousse [Jeuge-Maynart, 2010] : « Une donnée est le résultat d'observations ou d'expériences faites délibérément ou à l'occasion d'autres tâches et soumis aux méthodes statistiques. »

Dans un contexte de réseau informatique, elle peut être, par exemple, mesurée à travers des outils de supervision ou fournie par une personne utilisant le système. Citons les exemples suivants pour en comprendre le sens :

— Un opérateur de tripleplay a enregistré 100 tickets d'utilisateurs se plaignant d'une mauvaise qualité de la réception télévisuelle.
— Un serveur de diffusion de vidéo s'est éteint à 21h15 à la date du 10 octobre 2013 pendant 32 minutes.

1.7.3.1.2 Information Selon la définition du Larrousse [Jeuge-Maynart, 2010] : « L'information est tout événement, tout fait, tout jugement porté à la connaissance d'un public plus ou moins large, sous forme d'images, de textes, de discours, de sons. »

D'une manière plus générale, il s'agit d'une donnée à laquelle un contexte et

une interprétation sont fournis afin qu'un administrateur réseau puisse prendre la décision appropriée. Reprenons les exemples cités précédemment, les informations qui en découlent sont comme suit :

— La perception des usagers du service TV mesurée en MOS [1], est passée sous la barre des 3/5 cette semaine.
— La fiabilité du serveur est de l'ordre de 70% pour la semaine en cours.

Ces deux informations permettent à l'administrateur système de décider s'il est nécessaire d'apporter une action corrective si, bien évidement les perturbations ne sont pas le fruit d'événements extérieurs et non contrôlés comme une coupure électrique. À ce stade, il ne s'agit que d'une réaction ponctuelle, sans aucune considération à long terme.

1.7.3.1.3 Connaissance Selon la définition du Larrousse [Jeuge-Maynart, 2010] : « La connaissance est l'opération par laquelle l'esprit humain procède à l'analyse d'un objet, d'une réalité. » Un administrateur réseau réfléchirait à une situation au regard des informations à sa disposition ainsi qu'en fonction d'une expérience acquise avec le temps.

Typiquement, dans les exemples cités plus haut, l'opérateur aurait pu avoir une analyse plus fine : *L'insatisfaction des usagers est probablement dûe au manque de fiabilité du serveur de diffusion vidéo.* La corrélation de ces deux événements est essentielle pour apporter une solution rapide et adéquate, car, par expérience, l'administrateur sait qu'une insatisfaction prolongée conduirait, inéluctablement, à une augmentation du taux attrition [2] (Churn rate) et par conséquent, à une perte financière.

1.7.3.2 Construction des connaissances

Le processus de construction des connaissances commence par la collecte des données et des informations. En ce sens, Madhyastha et al. [2006] ont proposé une plateforme nommée iPlane (**i**nformation **Plane**). Elle consiste en un système distribué à travers le réseau mondial PlanetLab [Culler et al., 2002] qui effectue des mesures continues des métriques de qualité de service réseau (latence, délai, bande passante, taux de perte) dans le but de générer une carte annotée du réseau Internet (pour plus de détail, voir la figure 1.9

1. **M**ean **O**pinion **S**core est un score allant de 1 à 5 décrivant la perception de l'usager d'un service donné [Rec, 2006].
2. L'attrition désigne la perte de la clientèle.

tirée de [Madhyastha et al., 2006]).

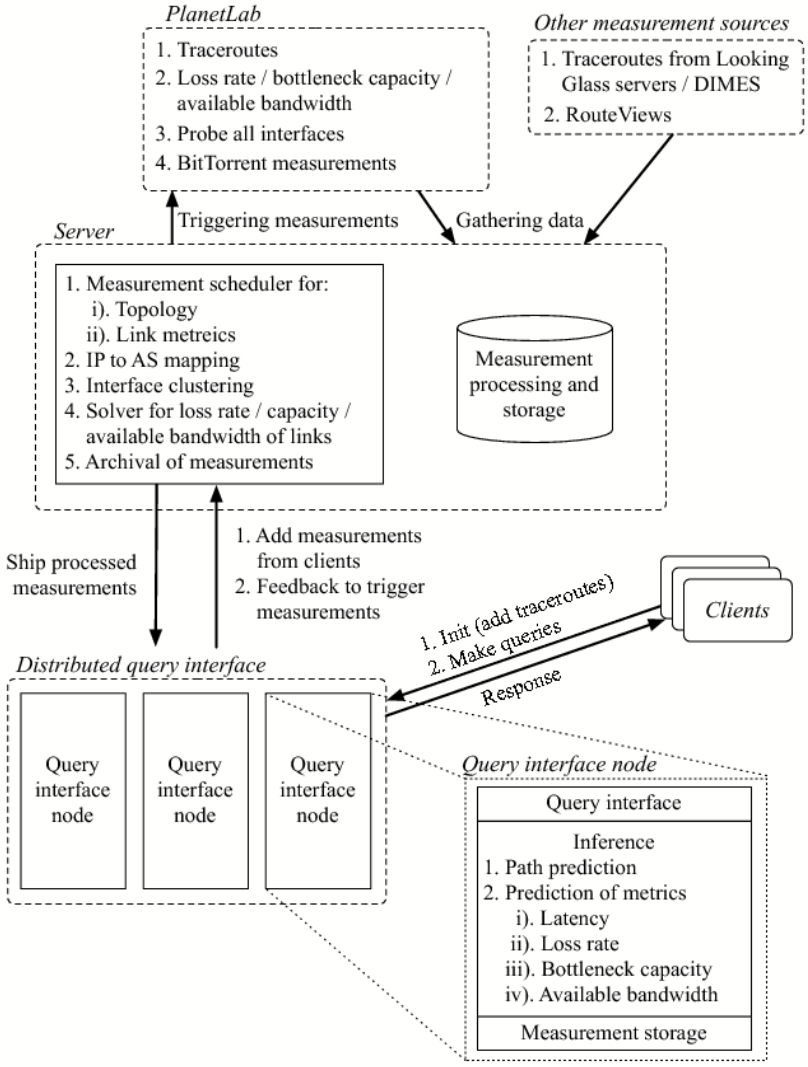


FIGURE 1.9 – Architecture de iPlane

Malgré le fait que iPlane utilise le terme information et non le terme de connaissance, il n'en demeure pas moins que certaines connaissances ont été déduites par corrélation entre les observations des différentes parties du réseau. L'ensemble de ses informations est stocké au niveau d'une base de données centralisée de type SQL-like.

Chen et al. [2007] proposent une autre approche d'évaluation des performances. Elle consiste à sélectionner un sous-ensemble de liens recouvrants et à évaluer leur taux de perte. Cette opération est répétée un nombre suffisant

de fois afin de pouvoir en déduire les taux de perte sur l'ensemble du réseau.

1.7.3.3 Réprésentation de la connaissance

Fournir une représentation de la connaissance dans le domaine des réseaux informatiques consiste à fournir une vue unifiée et évolutive (à défaut d'être exhaustive) de tous les concepts qui régissent ce domaine. Des organismes de standardisation ont proposé des ontologies et des modèles pour répondre à cette problématique :

MIB (Management Information Base) a été défini en 1991 par IETF [3] [McCloghrie and Rose, 1991]. Il consiste en une base d'informations sur les ressources du réseau géré.

Le Modèle CIM (Common Information Model) est un modèle conceptuel proposé par le DMTF [4] [DMTF, 1999]. Il permet de décrire les informations de gestion des équipements informatiques et des réseaux.

INDL Ontology (Infrastructure & Network Description Language) est une ontologie proposée afin de réponde à la nécessité d'un vocabulaire commun et partagé permettant de décrire les réseaux informatiques [Ham, 2010]. Cette ontologie est décrite dans le format *Resource Description Framework* (RDF) [Lassila et al., 1998].

Les nombreuses propositions faites dans ce cadre reflètent la difficulté de la tâche. En effet, vu le nombre de composants en vigueur, il n'est pas trivial d'identifier et de standardiser l'ensemble des connaissances pertinentes dans le réseau.

1.7.3.4 Dissémination de la connaissance

Le passage à l'échelle et la robustesse représentent des caractéristiques importantes qui doivent être prises en compte dans la construction d'un plan de connaissance. En effet, la distribution des connaissances au sein d'un réseau intégrant un nombre élevé d'équipements diversifiés ne peut pas se faire par des approches de type diffusion, ceci impliqueraient une surcharge importante du réseau. C'est pour cette raison que plusieurs approches de dissémination ont été proposées. Elles sont décrites ci-dessous :

3. Institute of Electrical and Electronics Engineers
4. Distributed Management Task Force

1.7.3.4.1 Diffusion locale La première solution consiste à propager les connaissances uniquement dans un voisinage immédiat. Les partisans de cette approche suggèrent que pour certaines architectures, il n'est pas nécessaire d'avoir une connaissance globale du système, mais uniquement une vue locale. Par exemple, dans [Tang et al., 2007], les auteurs proposent de diviser le réseau en zones géographiques ; les connaissances sont ainsi diffusées à l'intérieur de chaque zone. Dans [Schuetz et al., 2007], les auteurs proposent que chaque station de base synchronise ses connaissances avec les noeuds qui sont dans sa cellule de diffusion. De tels systèmes offrent une vue réduite des connaissances du réseau. Leurs performances dépendent donc du contexte d'utilisation.

1.7.3.4.2 Vue située La diffusion en vue située est décrite comme un nouveau concept de dissémination des connaissances [Nguengang et al., 2008 ; Bullot et al., 2008 ; Marrow and Manzalini, 2006]. Le terme "vue située" (Situatedness) est importé de la thématique multi-agents. Dans celle-ci, la vue située est une propriété des agents autonomes qui consiste à se situer dans son environnement, le percevoir et interagir avec lui [Florian, 2003].

Dans le cas du plan de connaissance, la vue située représente la limite que l'agent s'impose pour aller chercher ou diffuser une connaissance. Du fait que cette approche ne diffuse les connaissances que dans cette zone, elle permet un déploiement à large échelle.

L'inconvénient ici est celui de déterminer de manière distribuée la portée idéale de la vue située qui permet d'avoir le maximum de connaissances pertinentes sans surcharger le réseau. En effet, plus cette zone est importante, plus les connaissances sont globales et pertinentes, mais cela a comme inconvénient celui de générer un plus grand nombre de requêtes augmentant ainsi les temps de réponse. Toutefois, une zone trop petite conduira à une prise de décision inadaptée qui serait la conséquence de connaissances incomplètes.

1.7.3.4.3 Table de hashage distribuée Une table de hachage distribuée (ou DHT) [Stoica et al., 2001], est une technique qui permet de gérer une très grande base d'informations. Elle consiste à répartir la table de hachage [5] sur tous les éléments du réseau, qui en possèdent chacun une partie.

Par exemple, le noeud A va être responsable de toutes les connaissances qui commencent par A, de même que les noeuds B et C …. Lorsqu'un noeud souhaite récupérer une connaissance, il commence par rechercher le noeud qui

5. Une table de hashage est une structure de données de type clé-élément.

en est responsable avant de la demander.

De par sa simplicité de mise en oeuvre, cette approche est utilisée dans plusieurs travaux [Strassner et al., 2006 ; Li et al., 2008]. Toutefois, la nécessité de rechercher le détenteur de la connaissance et ensuite la récupérer pour son traitement fait que cette méthode n'est pas forcément adaptée à tous les cas d'usage, particulièrement dans le cas où le système nécessite un temps de réponse très court.

1.7.3.4.4 Cloud La proposition de stocker et de gérer le plan de connaissance dans le Cloud est assez nouvelle. Yu et al. [2012] proposent d'utiliser une "bigtable" [Chang et al., 2008] pour gérer les connaissances produites par un très grand ensemble de capteurs.

Une bigtable est un système de gestion de base de données distribuée, compressée et à haute performance, de type *Not only SQL* (NoSQL) [6], proposé par Google [Chang et al., 2008]. Les données sont modélisées sous la forme d'un couple clé/valeur. Elles sont stockées dans un système de fichiers (Google FS) dans un format propriétaire (SSTable : Sorted String Table).

Toutefois, le fait que les connaissances soient stockées dans un emplacement distant (datacenter), ajoute une certaine latence lors de leur récupération, ce qui peut ne pas convenir à certaines applications. De plus, le réseau n'est pas à l'abri d'une panne du lien le reliant avec la bigtable. Enfin, le stockage des connaissances chez un hébergeur tiers peut poser certains problèmes de confidentialité des données.

1.8 Conclusion

Les concepts liés à l'autonomique permet aux réseaux de s'auto-organiser à la manière du monde du vivant. Ce concept est avant tout un besoin des opérateurs pour gérer au moindre coût des réseaux qui deviennent de plus en plus complexes. Dans un premier temps, nous avons présenté ce concept ainsi qu'une étude de l'éxistant. De plus, nous avons mis en évidence la nécéssité d'un nouveau plan nommé "plan de connaissance" qui vient se rajouter aux trois plans traditionnels existants. L'émergence de ce plan a impliqué la définition de plusieurs aspects liés à la gestion de cette connaissance : sa dé-

6. NoSQL désigne une catégorie de systèmes de gestion de base de données qui ne sont pas fondés sur l'architecture des bases relationnelles.

finition, sa construction, sa représentation ou encore sa dissémination. Dans une seconde partie de ce chapitre, nous avons défini chacun de ses verrous et nous nous sommes attardés sur la problématique liée à la dissemination des connaissances sur les éléments du réseau.

Dans le chapitre suivant, nous présentons nos contributions dans le cadre de cette dissémination.

Chapitre 2

Approche adaptative et hiérarchique pour la dissémination des connaissances

Sommaire

2.1 Introduction

Une des caractéristiques principales du plan de connaissance est qu'il doit rendre toute nouvelle connaissance disponible aux noeuds qui s'y intéressent,

et cela, de la manière la plus optimisée possible. La solution la plus simple est, bien évidemment, de diffuser cette connaissance simultanément à l'ensemble des éléments du réseau, or, cette idée est difficilement réalisable dans la réalité. En effet, cette façon de faire engendre une surcharge importante sur les liens étant donné que le coût de diffusion dans le réseau augmente d'une façon exponentielle en fonction de la distance de diffusion [Nguengang et al., 2008]. Le challenge est donc de proposer une méthode de dissémination des connaissances qui répond, à la fois, aux caractéristiques énoncées dans [Clark et al., 2003] (cf. section 1.7.2) et avec un coût (au sens surcharge du réseau) contenu.

Afin de répondre à cette problématique, nous avons proposé un mécanisme de dissémination des connaissances se basant sur deux idées clés :

— Choisir un ensemble d'éléments du réseau (noeuds) en charge de la gestion et de la diffusion des connaissances. Ce choix devrait être adaptatif en fonction du contexte.
— Construire plusieurs instances de plans de connaissance où chaque instance est ensuite activée selon le contexte dans lequel se trouve l'état du réseau. Ce contexte dépend de plusieurs paramètres restant à définir et à modéliser.

2.2 Sélection des super noeuds

Afin de minimiser la surcharge des liens, nous proposons d'organiser le réseau dans une structure hiérarchique virtuelle composée de super noeuds et de noeuds normaux.

Un super noeud agit comme un serveur central pour un ensemble de noeuds normaux qui lui sont rattachés. Ainsi, la connaissance est gérée uniquement au niveau des super noeuds et les noeuds normaux y accèdent à travers les super noeuds auxquels ils sont rattachés. Ces super noeuds sont ensuite reliés entre eux au travers d'un réseau virtuel qu'il faudra définir. Le plan global de connaissance sera représenté dans ce cas par l'ensemble des bases de connaissance distribuées au niveau des super noeuds et sera donc accessible par tout noeud du réseau.

Bien que l'architecture à base de super noeuds ait été étudiée dans le cadre de la gestion des connaissances dans les réseaux autonomiques [Diaz and Chen, 2008 ; Abdeljaouad and Karmouch, 2012], le problème de sélection

de ces super noeuds demeure un problème ouvert. Ce problème a été défini par Lo et al. [2005] comme étant la sélection d'un ensemble de pairs dans un très grand réseau recouvrant (overlay network). Dans cette architecture, chaque noeud sera connecté à un super noeud comme le montre la figure 2.1. Les pairs sélectionnés fournissent un service particulier pour l'ensemble des pairs du réseau. La difficulté, ici, est qu'un grand nombre de noeuds peuvent être désignés comme super noeuds. Ce processus peut évoluer au cours de temps compte tenu de la forte dynamicité du réseau.

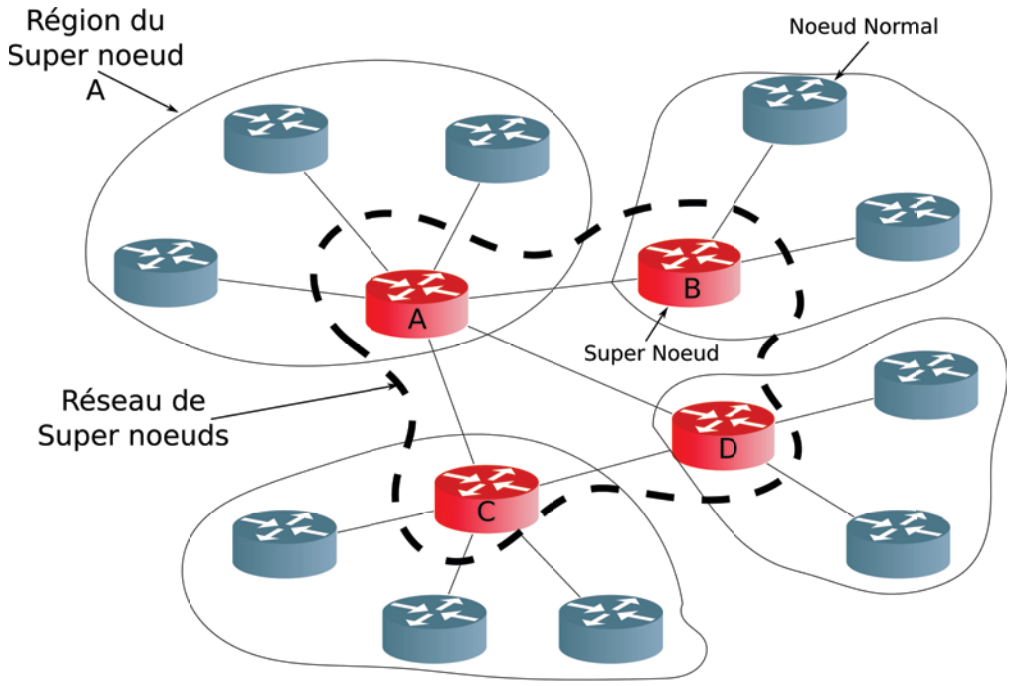


FIGURE 2.1 – Topologie d'une architecture hiérarchique à base de super noeuds

Nous allons, dans un premier temps, déterminer les critères clés de la sélection de ces super noeuds.

Ensuite, nous passons en revue les différentes propositions qui existent dans la littérature pour effectuer cette sélection.

Enfin, nous présentons un formalisme du problème posé ainsi que notre contribution pour le choix des supers noeuds.

2.3 Critères de sélection des super noeuds

Les super noeuds doivent être distribués sur l'ensemble du réseau afin de couvrir, au mieux, l'ensemble des noeuds qui le constitue. Lo et al. [2005] ont proposé un ensemble de contraintes que cette distribution doit respecter totalement ou partiellement. Nous les détaillons ci-dessous :

— L'accès : les noeuds non sélectionnés (ou noeuds normaux) doivent accéder à un ou plusieurs super noeuds avec un minimum de latence. Cette latence peut être mesurée par le délai aller-retour entre un noeud normal et un super noeud plus connu sous l'appellation anglo-saxone *Round Trip Time* (RTT).
— La dispersion : les super noeuds doivent être dispersés sur l'ensemble du réseau et non pas concentrés dans un sous-ensemble du réseau afin d'éviter des effets de goulot d'étranglement.
— La proportion : un ratio, préspécifié en fonction de l'application, de super noeuds, doit être maintenu afin de garantir de bonnes performances.
— Le partage de charge : un super noeud ne peut être en charge que d'un ensemble k de noeuds normaux. k est un facteur qui peut être défini en fonction de l'application ou en fonction des ressources matérielles disponibles.

En plus des quatre contraintes proposées dans [Lo et al., 2005], nous proposons d'en rajouter d'autres afin de rendre le mécanisme de sélection le plus efficient possible :

— La surcharge de messages de contrôle (ou Overhead) : afin de maintenir l'architecture hiérarchique, un échange de messages explicites est souvent nécessaire. Les échanges fréquents d'informations peuvent donc consommer considérablement la bande passante et nuire ainsi aux performances intrinsèques du réseau.
— Le cycle de calcul : Il s'agit du nombre de cycles nécessaires à l'accomplissement d'une procédure de sélection des super noeuds. En effet, tant que la sélection n'est pas effective, le réseau et les applications qui l'utilisent ne peuvent pas fonctionner correctement.

2.4 Définition

Le problème de sélection d'un super noeud peut être vu comme un problème de "leader election". Un représentant (ou leader) est un élément élu ou désigné capable de prendre une initiative au nom d'un groupe. Il joue le rôle d'interface pour les éléments ordinaires qui en dépendent. Dans le calcul distribué, l'élection du leader est le processus de désignation d'un seul et même processus organisateur de certaines tâches réparties entre plusieurs noeuds. À l'initialisation, aucun noeud du réseau ne connaît de leader. Un algorithme d'élection du leader est alors exécuté et à chaque noeud normal du réseau sera donc associé un leader. Les noeuds du réseau communiquent donc entre eux afin de décider lequel d'entre eux est le "leader". Pour cela, ils ont besoin d'un critère de sélection. Par exemple, si chaque noeud a une identité unique, les noeuds peuvent décider simplement que le noeud disposant de la plus grande identité soit le leader.

La définition de ce problème a été souvent attribuée à Lelann [Lann, 1977], reprise ensuite en tant que méthode pour créer un nouveau jeton dans un réseau en anneau dont le jeton a été perdu. Ce problème devient encore plus ardu s'il est nécessaire d'élire n Leaders dans un groupe en respectant certaines contraintes liées à l'application. Un tel problème peut être défini comme un problème d'ensemble dominant.

Dans la théorie des graphes, un ensemble dominant d'un graphe $G = (S, A)$ est un sous-ensemble S' tel que tout sommet de l'ensemble S possède au moins une arrête commune avec un sommet de S'. Ce problème a été décrit par Haynes et al. [1998] comme un problème NP-complet.

De nombreuses heuristiques développées pour des problèmes classiques ont été utilisées pour la sélection des super noeuds dans le réseau, mais leur utilité se limite aux réseaux faiblement dimensionnés avec peu de variabilité au cours du temps (exemple, les algorithmes à coloriage [Guha and Khuller, 1998]).

2.5 Les approches de sélection des super noeuds

Pour résoudre ce problème, plusieurs approches existent. Nous les avons classé en plusieurs catégories comme le montre le tableau 2.1. Les approches simples choisissent les super noeuds selon un critère de seuil. Ce choix est

généralement figé dans le temps. Ainsi, un super noeud choisi ne cédera sa place à un meilleur candidat qu'en cas de panne. Les approches adaptatives permettent de faire évoluer l'ensemble des super noeuds dans le temps.

Approches adaptatives	Approches simples	
	Choix à seuil	Choix centralisé
SG-1 Montresor [2004]	KaZaA [Liang et al., 2004]	SPSP [Wolf and Merz, 2007]
SG-2 [Jesi et al., 2006]	Gnutella [Garbacki et al., 2010]	
SPSA [Merz et al., 2008]	Skype [Baset and Schulzrinne, 2006]	

TABLE 2.1 – Systèmes de sélection des super noeuds

2.5.1 Approches simples

Dans cette section, nous décrivons 3 systèmes qui se basent sur la notion de seuil ainsi qu'un algorithme dans lequel la méthode de sélection est centralisée. Ces méthodes sont, à nos yeux, intéressantes à étudier, car ils montrent l'intérêt des architectures à base de super noeuds dans des cadres applicatifs.

KaZaA est le premier réseau pair-à-pair de partage de fichiers qui introduit les architectures à base de super noeuds [Liang et al., 2004] pour améliorer la fonction de recherche. Les clients sont divisés en 2 classes :

- Ceux avec une grande capacité sont définis comme étant des super noeuds. Ces noeuds maintiennent un index des fichiers stockés par les clients.
- Les autres sont des clients normaux. Ils soumettent les requêtes de recherche aux super noeuds.

Cette architecture permet à KaZaA d'améliorer la rapidité de la recherche, de réduire son coût et d'assurer la faisabilité de l'approche pour un grand nombre de noeuds. Toutefois, il est difficile de déterminer les seuils à partir desquels un client est considéré comme super noeud. D'une part, un seuil trop bas ferait que tous les clients seront des super noeuds, nous ramenant ainsi à une architecture sans hiérarchie et, par conséquent, inopérationnelle. D'autre part, un seuil trop élevé donnerait lieu à des super noeuds surchargés générant ainsi un problème de goulot d'étranglement.

Gnutella est un protocole pair-à-pair où chaque noeud est considéré à la fois comme client et comme serveur [Ripeanu, 2001]. Une topologie virtuelle relie les noeuds entre eux. La procédure de recherche consiste à interroger les voisins en leur adressant une requête. Celle-ci est ainsi transmise de voisin à voisin tant que sa durée de vie n'est pas écoulée. Cette inondation coûteuse a conduit à faire évoluer Gnutella vers une architecture à base de super noeuds (appelés ultrapeers) [Garbacki et al., 2010]. Ces ultrapeers sont les seuls noeuds impliqués dans une recherche par inondation. Pour prétendre devenir un ultrapeer, un noeud doit avoir des débits montants et descendants conséquents.

L'architecture des ultrapeers a grandement amélioré la scalabilité de Gnutella. Toutefois, l'inconvénient majeur de Gnutella, comme KaZaA, reste le problème des seuils. En effet, avec les débits actuels, les seuils préconisés par la version 0.6 de Gnutella feraient que l'ensemble des noeuds du réseau peuvent prétendre être des super noeuds [Garbacki et al., 2010]. Cependant, en prenant en considération les avancés technologiques, il faut en permanence redéfinir manuellement ces seuils.

Skype Skype est une solution de téléphonie IP pair-à-pair [Baset and Schulzrinne, 2006]. Elle utilise aussi les super noeuds pour relayer les communications dans le réseau Internet. Pour qu'ils puissent fonctionner comme un relais, ces noeuds doivent être accessibles à travers Internet (il ne faut pas qu'ils soient bloqués par un pare-feu ou dans un réseau avec translation d'adresse). Toutefois, rien ne garantit que ces noeuds, sélectionnés uniquement sur le critère d'accessibilité, soient les plus adaptés pour relayer la communication. En effet, ces noeuds peuvent être surchargés ayant pour conséquence une détérioration de la session.

SPSP (Super-Peer Selection Problem) consiste à trouver les super noeuds en utilisant une connaissance complète du réseau [Wolf and Merz, 2007]. L'algorithme commence par la sélection aléatoire de p noeuds qu'il considère comme superpairs. Tous les autres pairs sont affectés au plus proche superpair sur la base du calcul du délai d'aller-retour. Si la topologie créée viole une contrainte (délai d'aller-retour) fixée a priori, le processus est répété. Au cours de l'évolution du processus, les pairs sont affectés à d'autres superpairs et les superpairs peuvent devenir des pairs normaux. La solution appliquée ici n'est pas optimale ; elle ne garantit pas en effet un partage de charge entre les superpairs. De plus, le seuil de violation de la contrainte est difficile à fixer au regard de la dynamique intrinsèque du réseau correspondant aux types de trafic et aux changements de topologie.

Bien que peu adaptées à notre problématique, en raison de la difficulté du choix des seuils et de l'impossibilité de garantir la qualité de l'ensemble des noeuds sélectionnés, les approches simples contribuent à montrer l'importance des architectures à base de super noeuds. De plus, l'approche SPSP Globale [Wolf and Merz, 2007], qui propose un ensemble de super noeuds respectant une contrainte fixée a priori, nous servira dans nos évaluations futures comme point de comparaison pour les approches distribuées que nous proposons.

2.5.2 Approches adaptatives

Dans cette section, nous détaillons 3 algorithmes d'élection et d'optimisation d'un ensemble de super noeuds dans le cas d'approches adaptatives. Dans SG-1, le critère d'optimalité à atteindre est lié au respect de la capacité maximale des super noeuds. SG-2, qui constitue une amélioration de SG1, intègre, de plus, les distances entre les noeuds. Quant à la méthode SPSA (Super-Peer Selection Algorithm), elle constitue une variante distribuée de SPSP qui prend en compte le délai d'aller-retour entre les noeuds du réseau et leur super noeud respectif.

2.5.2.1 SG1

L'algorithme SG1 a été proposé par Montresor [2004] afin de générer une topologie à base de super noeuds répondant aux caractéristiques suivantes :

i) Chaque noeud doit être associé exactement à un seul super noeud,
ii) Les super noeuds sont reliés à travers un réseau recouvrant,
iii) Le nombre des super noeuds est minimisé.

Dans SG-1, chaque noeud n est défini avec une capacité C_p, qui détermine le nombre maximum de clients qu'il peut gérer s'il est élu comme super noeud. SG-1 considère que cette capacité est invariante dans le temps. L'approche mise en oeuvre dans SG-1 cible une topologie dans laquelle chaque noeud est rattaché à un super noeud avec une capacité des super noeuds au moins égale au nombre des noeuds du réseau.

Pour maintenir la topologie mise en place, SG-1 se base sur un algorithme dit de "chuchotement". Chaque noeud échange avec un certain nombre de ses voisins sa capacité résiduelle, le nombre de clients qu'il gère (s'il est super noeud) et ses voisins immédiats. Le but est de pouvoir échanger les rôles à

chaque cycle (intervalle de temps régulier) : un super noeud peut devenir un noeud normal au profit d'un autre noeud et l'inverse. Ainsi, il est aisé de migrer les clients des super noeuds à faible capacité vers les super noeuds à plus forte capacité et, donc, éliminer les super noeuds superflus (ceux qui n'ont plus de clients).

Le principal défaut de SG-1 est de ne pas prendre en considération la latence entre les super noeuds et les noeuds normaux. Ceci a été corrigé dans SG-2.

2.5.2.2 SG-2

L'algorithme SG-2 a été proposé par Jesi et al. [2006] comme une amélioration de SG-1. En plus de considérer la capacité des super noeuds lors de la phase de sélection, SG-2 considère aussi la latence.

Pour construire la topologie du réseau recouvrant des super noeuds, Jesi et al. [2006] décrivent le problème comme un problème géométrique. Chaque noeud du réseau est considéré comme un point dans un espace à 2 dimensions. Chaque point a une zone circulaire d'influence et l'objectif est de couvrir cet espace avec le minimum de points. Le réseau est dit couvert quand chaque point du réseau est, ou un super noeud, ou se situe dans la zone d'influence d'un super noeud. Si un point se trouve dans deux zones d'influence différentes, il choisit celle du noeud avec la plus grande capacité résiduelle.

Cet algorithme reste sensible à la dimension donnée à la zone d'influence. À cet effet, un algorithme de diffusion généralisée est utilisé pour maintenir la topologie. Chaque noeud envoie des requêtes périodiques dans sa zone d'influence. Plus la zone est grande, plus l'algorithme est consommateur en ressources. De même, plus la zone est petite, moins le résultat obtenu est fiable. Il n'est donc pas trivial de trouver la bonne distance de diffusion, rendant ainsi SG-2 difficile à mettre en place sauf dans des cas particuliers comme les réseaux sans fil. La zone ici peut être la cellule radio de chaque noeud.

2.5.2.3 SPSA

L'algorithme réparti SPSA [Merz et al., 2008] a été conçu pour construire une topologie à base de super noeuds sans qu'un noeud n'ait besoin d'une vision globale du système. Son fonctionnement consiste à effectuer régulièrement un cycle de réorganisation. Celui-ci diffère selon la nature des noeuds.

Durant ce cycle, un noeud simple peut décider d'être rattaché à un autre super noeud s'il en trouve un plus proche (en termes de délai d'aller-retour). De plus, si la connexion avec son super noeud est perdue, il peut alors contacter le super noeud qui lui est le plus proche. Un super noeud peut, quant à lui, effectuer 3 différentes actions :

— Il renonce à son rôle (déclassement) en faveur d'un noeud normal s'il constate le nombre de noeuds qui lui sont rattachés est faible (inférieur à la moitié du nombre des super noeuds qu'il connaît).
— Si ce nombre est supérieur à deux fois le nombre des super noeuds qu'il connait, il décide de promouvoir un noeud parmi ceux qui lui sont rattachés comme nouveaux super noeuds.
— Il échange les rôles avec un noeud qui lui est rattaché s'il constate que ce noeud est plus adapté que lui pour jouer le rôle de super noeud.

Le principal inconvénient de cette approche est lié aux décisions de déclassement et de promotion des noeuds. Ces dernières sont prises par chaque noeud en fonction des informations dont il dispose sur le réseau et sans aucune considération sur leur impact sur la topologie globale. Comme ces informations forment une vue locale du réseau (dans un voisinage donné), les décisions prises s'approcheront ainsi, au mieux, d'un optimum local. Par conséquence, il est probable que certains noeuds, se retrouvant desservis, se déclarent eux même super noeuds dans une zone où ils sont tous seuls ou encore rejoignent un super noeud éloigné. Dans les deux cas de figures, cela détériore la topologie construite.

2.5.2.4 Discussion

Les approches adaptatives décrites ci-dessus ont été proposées pour répondre à des besoins spécifiques (réseau de capteur sans fil, application de transfert de fichier pair-à-pair, …). Néanmoins, nous les jugeons inadaptées à notre plateforme. En effet, même si SG-2 améliore la proposition faite par SG-1 en prenant en considération la latence, sa mise en application reste assez compliquée compte tenu de la difficulté à trouver la bonne distance de diffusion. Par ailleurs, SPSA repose sur une vision locale du réseau pour trouver les super noeuds, la solution auquelle il converge reste sous optimale du point de vue global.

Nous nous proposons de formaliser le problème de la construction de la topologie hiérarchique, consistant à élire un ensemble de noeuds responsables de la gestion des connaissances comme un problème de partitionnement K-

medoïds

2.6 Nouvelle défintion du problème : problème de partitionnement K-medoïds

Le partitionnement [Soni and Ganatra, 2012] consiste à découper un ensemble de points (dans notre cas, il s'agit de noeuds du réseau) en classes en minimisant une fonction objective donnée. Il peut être classifié en deux types : le partitionnement K-moyennes et le partitionnement K-medoïds.

Dans le partitionnement K-moyennes, chaque classe est représentée par un point virtuel qui se représente comme une moyenne calculée, éventuellement pondérée, des points qui la composent. Ces points virtuels sont appelés centroïds. Dans le partitionnement K-medoïds, une classe est représentée par un point réel d'une classe donnée. Ce point est appelé medoïd. Les classes sont composées d'un sous-ensemble de points à proximité d'un medoïd. Comme nous nous intéressons à trouver des points réels, le partitionnement K-medoïds est donc plus adapté à notre problématique.

Le problème de partitionnement peut être formalisé comme suit :

> Étant donné un ensemble de N objets caractérisés par un ensemble de propriétés (particularités, identifiants, ...), le but est de les regrouper dans des sous-ensembles significatifs.

En règle générale, un bon algorithme de partitionnement tend à affecter les entités similaires à la même partition et les autres entités aux partitions restantes. Il est donc capital de bien définir la notion de similarité et de dissimilarité.

2.6.1 Mesure de similarité et de dissimilarité

Une des plus importantes notions qui définit un processus de partitionnement est de bien déterminer les mesures de similarité et dissimilarité entre les objets à traiter. Ces notions peuvent être définies comme suit :

Soit N le groupe d'objets à partitionner. La mesure de similarité est définie par la fonction [Nakache and Confais, 2004] :

$$
\begin{array}{l}
similarity: \ N - \times - N \ \rightarrow \ [0,1] \\
\text{où :} \\
\forall \ x, y \ in \ N \\
0 \leq similarity \ (x,y) \leq 1 \\
similarity \ (x,x) \ = \ 1 \\
similarity \ (x,y) \ = \ similarity \ (y,x)
\end{array}
\tag{2.1}
$$

La variable $similarity\ (x, y)$ définit le degré de similarité entre les objets x et y : plus cette valeur est grande, plus x est dit proche de y.

De la même manière, la mesure de la dissimilarité entre deux objets est exprimée par la fonction [Nakache and Confais, 2004] :

$$
\begin{array}{l}
dissimilarity: \ N - \times - N \ \rightarrow \ [0,1] \\
\text{où :} \\
\forall \ x, y \ in \ N \\
0 \leq dissimilarity \ (x,y) \leq 1 \\
dissimilarity \ (x,x) \ = \ 0 \\
dissimilarity \ (x,y) \ = \ dissimilarity \ (y,x)
\end{array}
\tag{2.2}
$$

La variable $dissimilarity\ (x, y)$ reflète le degré de différence entre les objets x et y : plus cette valeur est grande, plus x est dit distant de y.

2.6.2 Partionnement k-medoïd

Le problème de partitionnement est décrit comme une méthode d'apprentissage non supervisée. Les données sont fournies sans qu'on n'ait une idée de la nature des classes désirées auxquelles elles appartiennent. Il s'agit d'un problème difficile d'un point de vue combinatoire dans la mesure où le nombre de possibilités augmente significativement avec l'augmentation du nombre d'entités à traiter. Il faut donc trouver une manière efficace de partionner les entités au regard des contraintes imposées par l'application. En effet, les méthodes utilisées pour partionner les entités dépendent fortement du domaine d'application. Pour un même ensemble d'objets, le résultat du partitionnement peut être totalement différent selon l'algorithme employé ou les mesures de similarité/dissimilarité retenues. De plus, le nombre de clusters obtenus à la sortie de l'algorithme peut être prédéfini a priori par l'utilisateur ou défini par l'algorithme lui-même en fonction des données à traiter.

Le partionnement K-medoïds consiste à trouver k classes $C_1, C_2, \ldots, C_k$ pour un ensemble de n objets en minimisant un ensemble de critères. L'un des critères les plus utilisés est de minimiser la valeur suivante :

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d\,(x, m_i) \tag{2.3}$$

où m_i représente le medoïd de la classe C_i et $d\,(x, m_i)$ la mesure de dissimilarité entre l'objet x et le medoïd m_i.

La stratégie de partitionnement est itérative. Elle consiste, dans ce cas, à construire k classes initiales et de changer les objets d'une classe à une autre afin de minimiser le critère E. Dans ce qui suit, nous allons décrire les principaux algorithmes de résolution de ce problème.

2.6.3 PAM

Partitioning Around Medoïds (PAM) est un algorithme de type K-medoïd [Soni and Ganatra, 2012]. Cette méthode sélectionne k objets représentatifs (medoïds) dans un ensemble d'objets donnés $x_1, x_2, \ldots, x_n$. Elle assigne chacun de ses objets au plus proche medoïd. Les objets représentatifs doivent être le centre des classes définies. La qualité du partitionnement est mesurée par la moyenne de la dissimilarité entre chaque objet et son medoïd. La procédure de sélection est représentée par l'algorithme 1.

Algorithme 1 L'algorithme PAM : Partitioning Around Medoïds

```
1 : N noeuds, un entier k.
2 : Init : Sélectionner aléatoirement k noeuds et les définir comme medoïds.
3 : Associer chaque objet au medoïd le plus proche.
4 : pour chaque medoïd m faire
5 :   pour chaque non-medoïd r faire
6 :     Échanger m et r
7 :     Calculer la dissimilarité totale de la nouvelle configuration.
8 : Sélectionner la configuration avec la dissimilarité totale la plus faible..
```

Dans notre cas, la mesure de dissimilarité concerne le calcul du délai aller-retour entre deux objets du réseau. Ce délai est mesuré régulièrement au travers de capteurs actifs. Notre objectif ici est de réduire le temps d'accès de chaque objet du réseau à la base de connaissance gérée par son super noeud

(medoïd) correspondant de manière à rendre la connaissance disponible au niveau de l'objet aussi vite que possible.

Le résultat obtenu par l'algorithme PAM est généralement de bonne qualité. Cependant, il a un temps d'exécution très lent pour les grands ensembles d'objets et de ce fait, est peu adapté à notre problématique. De plus, comme PAM est un algorithme centralisé, on retrouve ici les inconvénients de ce type d'algorithme : manque de robustesse (panne de l'élément central), inadéquation pour un passage à l'échelle (problème de goulot d'étranglement), ….

2.6.4 CLARA

Clustering LARge Applications (CLARA) est une méthode de partionnement proposée par Kaufman and Rousseeuw [2008] qui permet de traiter un très grand ensemble d'objets. Inspirée de PAM, CLARA consiste à appliquer le principe de partionnement sur un sous-ensemble d'objet au lieu de l'intégralité de l'ensemble des objets. Cette procédure est répétée de manière itérative afin de trouver une solution qui satisfasse le mieux la fonction objective globale consistant dans la minimisation de la somme des dissimilarités entre chaque medoïd et les objets qui lui sont rattachés.

Différentes applications ont montré que l'algorithme utilisé dans CLARA est performant en termes de qualité de partitionnement obtenu et de temps d'exécution. Cependant, CLARA reste un algorithme centralisé. Par ailleurs, les classes construites ne cont pas forcément équilibrées. Ceci ne répond pas aux objectifs de notre plateforme. Nous rappelons ici que nous cherchons une méthode de partionnement distribuée devant assurer à la fois une charge équilibrée entre les noeuds médoids et une dissimilarité faible.

2.7 Approches proposées

2.7.1 Proposition 1 : algorithme de partionnement k-medoïd distribué

Notre première contribution, nommée DCLARA, consiste à proposer un algorithme K-medoïd distribué pour la recherche d'un ensemble de super noeuds potentiels. Cet algorithme, inspiré de CLARA, nécessite un découpage préa-

lable du réseau en zones [1]. Les détails de l'approche proposée sont décrits dans l'algorithme 2.

Algorithme 2 Sélection des Super noeuds

```
p : la probabilité d'être super noeud.
INIT :
Chercher un super noeud dans le voisinage
si réponse positive alors
    Rejoindre le super noeud
sinon
    Tirer un nombre aléatoire n entre 0 et 1
    si n > 1 − p alors
        Aller "à SUITE"
    sinon
        Aller à "INIT" au delà d'un temps aléatoire t
SUITE :
Se déclarer comme super noeud dans le voisinage
Partager la connaissance avec les autres super noeuds de la zone.
si le noeud possède l'adresse IP la plus petite (en comparaison binaire)
  entre les super noeuds de la zone alors
    Se déclarer comme hyper noeud
    Partager la connaissance avec les autres hyper noeuds
    Executer CLARA
    Désigner de nouveau les Super noeuds et l'hyper noeud de la zone
    Partager les connaissances avec les super noeuds de la zone
    si le noeud est déclassé par CLARA (ne fait plus partie de l'ensemble
      des noeuds sélectionnés) alors
        Prévenir les noeuds dont il est responsable.
        Revenir à "INIT"
```

Chaque noeud qui intègre le réseau commence par envoyer une requête à ses voisins directs en quête d'un super noeud. Tous les autres noeuds peuvent y répondre avec des informations sur leur super noeud respectif. Si le noeud à l'origine de la requête reçoit une réponse, il adhère à la zone gérée par le super noeud ayant répondu à sa requête sinon il peut devenir un super noeud avec une probabilité p (p sert à sélectionner aléatoirement des super noeuds dans l'étape d'initialisation). Cette phase se répète chaque période d'une durée

1. On définit une zone comme un sous-ensemble de noeuds du réseau de petite taille. Une zone peut-être définie par un administrateur ou déduite par les noeuds eux-mêmes. Un exemple de zone est l'ensemble de 254 machines dans la même plage d'adresse IP de classe C.

aléatoire t. Elle se termine si le noeud est contacté par un super noeud ou s'il l'est devenu lui même entre-temps. Cela assure ainsi aux noeuds isolés de pouvoir adhèrer à une zone gérée par un super noeud.

Tous les super noeuds font savoir qu'ils ont ce statut à leur voisinage. Les noeuds qui reçoivent une requête d'invitation consistant à rejoindre la zone d'un super noeud, et qui ne sont pas eux-mêmes des super noeuds, mettent à jour leur super noeud référent et commencent à lui envoyer les connaissances collectées.

Chaque super noeud diffuse les connaissances en sa possession aux autres super noeuds. Celui disposant de l'adresse IP la plus petite (en comparaison binaire, par exemple, 10.0.0.3 est plus petit que 10.4.0.3) dans chaque zone, est désigné pour être hyper noeud. Ce dernier est ensuite chargé d'exécuter l'algorithme CLARA. Les hyper noeuds se partagent par ailleurs leurs connaissances respectives ; il est en effet nécessaire que chacun d'entre eux ait une vision globale des paramètres du réseau sans pour autant exiger une cohérence forte au niveau de la base de connaissance. En effet, un changement minime dans les paramètres du réseau dans une zone éloignée n'a pas d'influence, au niveau local, sur le résultat d'exécution de CLARA. Chacun de ces hyper noeuds répercute le résultat de l'exécution de CLARA uniquement dans sa zone. Le cloisonnement des zones permet d'éviter les incohérences dans la topologie créée. En effet, aucun noeud ne reçoit deux requêtes d'adhésion à deux zones gérées par deux super noeuds différents. La figure 2.2 montre le déroulement de cet algorithme.

Au delà de la sélection et dans l'objectif d'équilibrer la charge entre tous les super noeuds du réseau, nous proposons l'algorithme 3 décrit ci-dessous.

Si un super noeud a atteint son quota défini a priori (nombre de noeuds qu'il peut gérer) lors de la phase d'affectation et qu'il existe encore un noeud orphelin, qui devrait lui être rattaché, il intègre celui-ci et démarre une procédure d'équilibrage. Celle-ci consiste à chercher, parmi les noeuds qu'il gère, celui qui sera le moins affecté en termes de temps aller-retour (temps d'accès aux connaissances gérées par le super noeud) par un changement de super noeud (y compris le nouveau noeud à affecter) et l'affecte à son second plus proche super noeud. La figure 2.3 synthétise le déroulement d'une telle situation.

Comme le choix des super noeuds se fait de manière permanente, cela pose un problème pour des réseaux à forte dynamicité. En effet, un choix permanant de super noeuds conduit à une dégradation progressive du mécanisme de

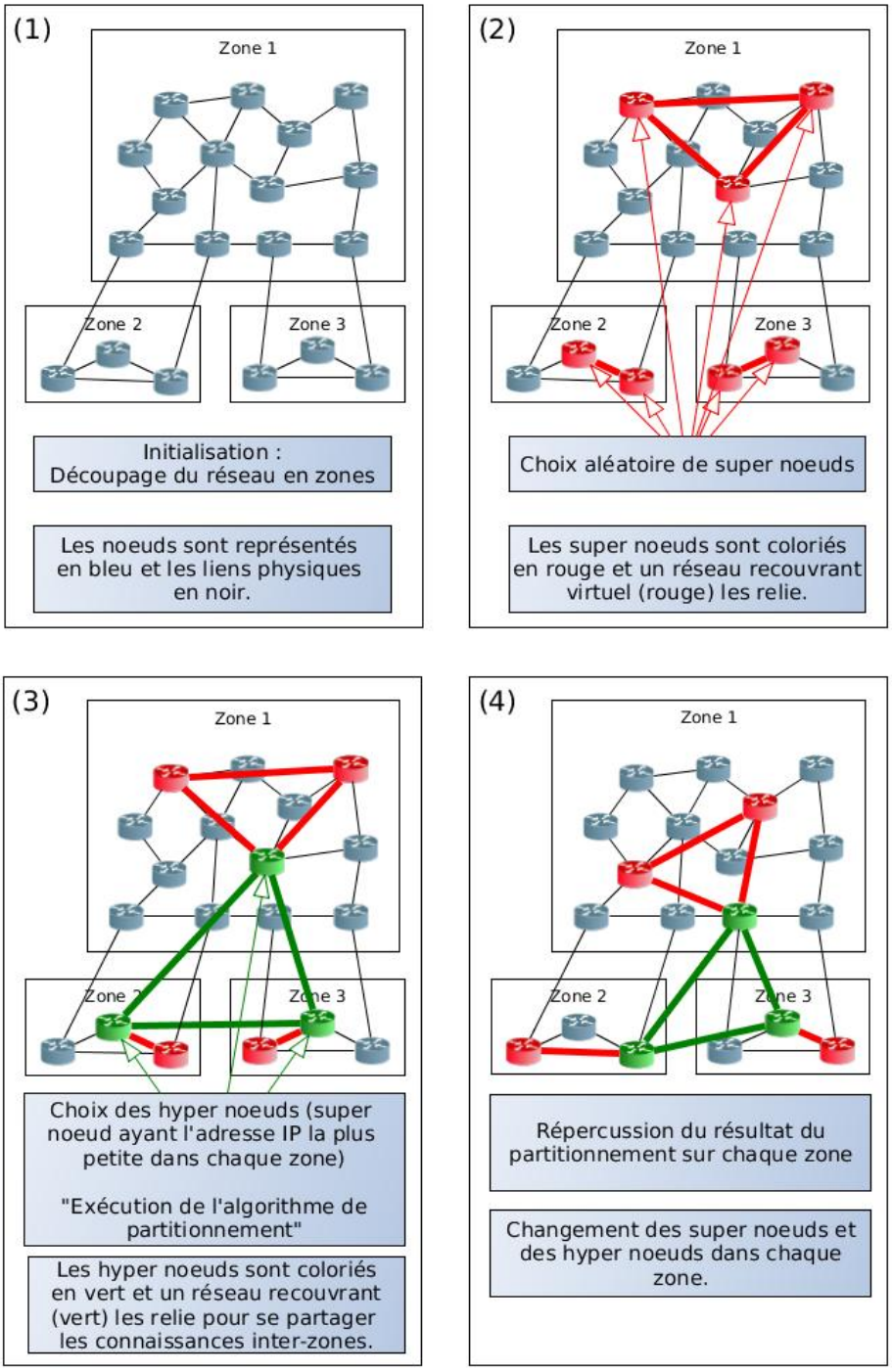


FIGURE 2.2 – Déroulement de l'algorithme de sélection des super et des hyper noeuds

Algorithme 3 Équilibrage de charges

```
1 : si Quota du super noeud s1 atteint et il existe encore un noeud orphelin alors
2 :     Prendre en charge le noeud
3 :     Calcul du préjudice (différence de la distance (mesure du délai aller-retour) séparant un noeud de ses deux plus proches super noeuds) pour tous les noeuds gérés
4 :     Choisir le noeud n1 avec le plut petit préjudice
5 :     Demander à son second super noeud potentiel s2 de le prendre en charge

6 :     s2 envoie une notification à n1
7 :     n1 change son super noeud référant
8 :     s1 enlève n1 de la liste des noeuds gérés
```

diffusion des connaissances (augmentation de la latence lors de la récupération des connaissances et du temps de dissémination de celles-ci). Pour éviter une telle dégradation, un processus de re-partitionnement est nécessaire. Cependant, cette réorganisation des noeuds du réseau reste une opération coûteuse et nécessite la mise place d'un critère mesurable pour son lancement. Nous décrivons dans la suite de ce chapitre la méthode que nous avons développée dans ce cadre.

2.7.2 Proposition 2 : algorithme adaptatif

Une façon simple de s'adapter au caractère dynamique du réseau et aux multiples changements de ses paramètres consiste à effectuer des opérations répétées à intervalles de temps réguliers conduisant à son re-paritonnement. Une telle opération peut aussi être enclenchée si un seuil de dégradation (une augmentation de la mesure de dissimilarité totale dans le réseau par exemple), fixé a priori, est atteint. Le choix de tels paramètres dépend fortement de la nature de l'application (quel serait par exemple le délai de latence admissible par une application donnée lors de la récupération d'une connaissance ?).

Nous avons opté dans le cadre de notre plateforme pour une sorte de re-partionnement adaptatif corrélé aux changements qui peuvent impacter la topologie physique du réseau (panne, ajout de lien ou de noeud, saturation de certains liens, …). Le lancement automatisé d'une telle procédure interviendra à tout changement d'état dans le réseau. Pour cela, un tel changement doit être détecté. Ceci constitue un problème particulièrement important dans la

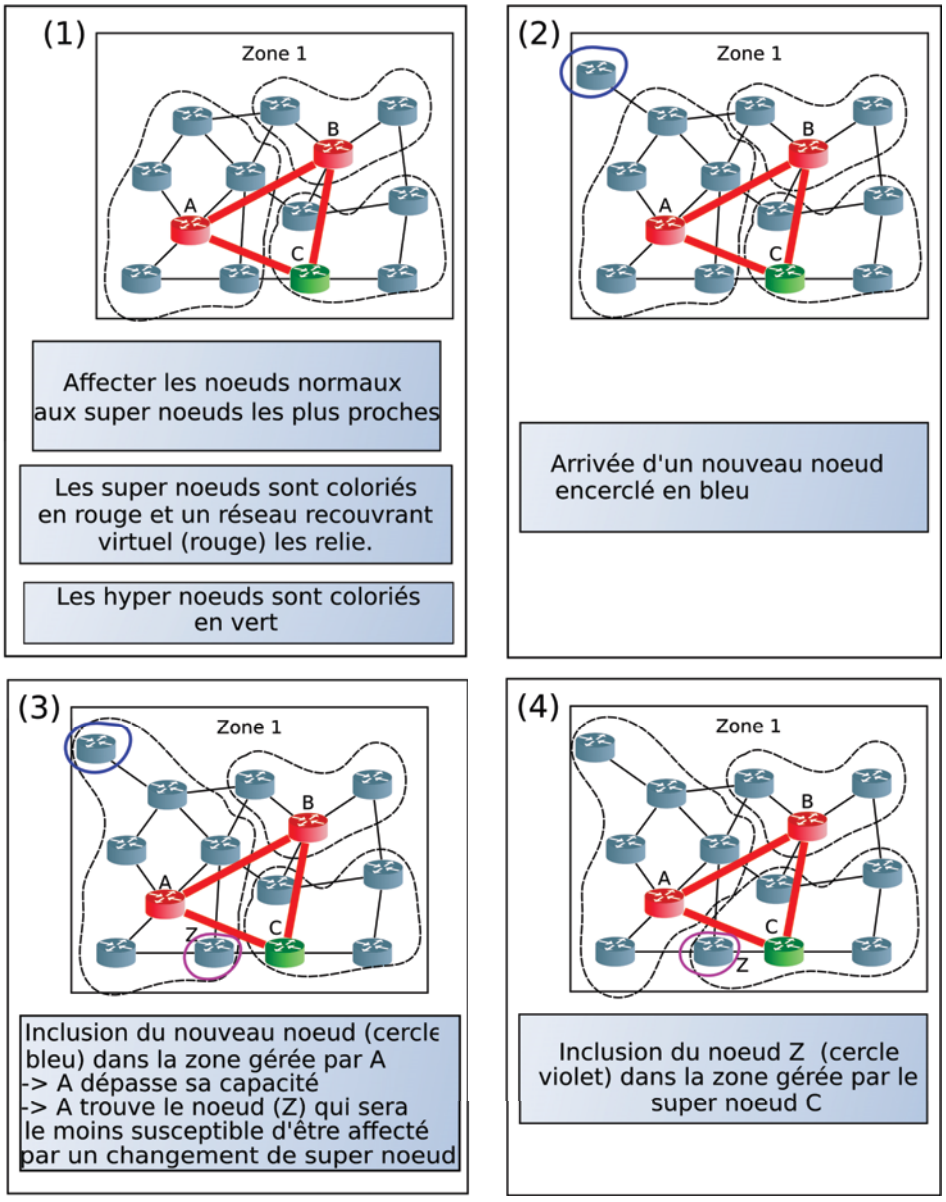


FIGURE 2.3 – Déroulement de l'algorithme d'équilibrage des charges

détection de défauts dans le fonctionnement d'un système. En effet, c'est précisément au cours d'un changement de nature imprévisible que se concentre une part prépondérante de l'information collectée pouvant servir ensuite à faire de la prévention. Pour ce faire, nous proposons une nouvelle approche (notée DCLARA-PH) utilisant le test statistique de Page-Hinkley (PHT) [Sebastião and Gama, 2009] pour détecter ces changements dans le comportement normal d'un système ne nécessitant aucune hypothèse au préalable. Comme

décrit dans l'équation 2.4, ce test évalue une variable cumulative X_t, définie comme la différence cumulée entre les valeurs observées à des instants donnés (d_t) et leurs moyennes ($\bar{d_T}$). σ est l'amplitude de changement autorisée (l'intevalle de tolérance).

$$\begin{array}{l} X_t = \sum_{t=1}^{T} d_t - \overline{d_T} - \sigma \\ \overline{d_T} = 1/T \sum_{t=1}^{T} d_t \\ \sigma : \text{l'amplitude de changement autorisée} \end{array} \tag{2.4}$$

Comme décrit dans l'algorithme 4 ci-dessous, le test consiste en la surveillance de la variation (notée $dPHT$) de la différence cumulée entre les valeurs observées à des instants donnés (mesure du délai aller-retour entre les noeuds et leur super noeud) et leur moyenne.

Algorithme 4 Test de Page-Hinkley

Matrice de distance (mesure du délai aller-retour) $d_1..d_t$
Valeur de σ amplitude de changement autorisée
Valeur de λ seuil de détection de changement d'état.
pour $t > 0$ **faire**
  $\bar{d_T} = 1/T \sum_{t=1}^{T} d_t$: moyenne des valeurs observées
  $X_t = \sum_{t=1}^{T} (d_t - \bar{d_T} - \sigma)$: différence cumulée des valeurs observées et leur moyenne.
  $m_T = min(X_t, t = 1..T)$: la plus petite différence observée
  $dPHT = X_t - m_T$
  si $dPHT > \lambda$ **alors**
    Continuer
Changement d'état détecté à l'instant t

Lorsque la variation $dPHT$ est supérieure à un seuil de détection λ, fixé a priori (expérimentalement), un changement d'état est détecté. λ impacte directement le taux des fausses alarmes observées (détecter un changement d'état alors qu'il n'en est rien) : plus la valeur de λ est élevée, plus le pourcentage de faux positifs (demande d'un re-partionnement alors qu'il n'est pas nécessaire) est faible. En conséquence, ce phénomène influe sur la surcharge générée par l'approche proposée et sur le délai de détection du changement d'état. La figure 2.4 montre le déroulement de cet algorithme.

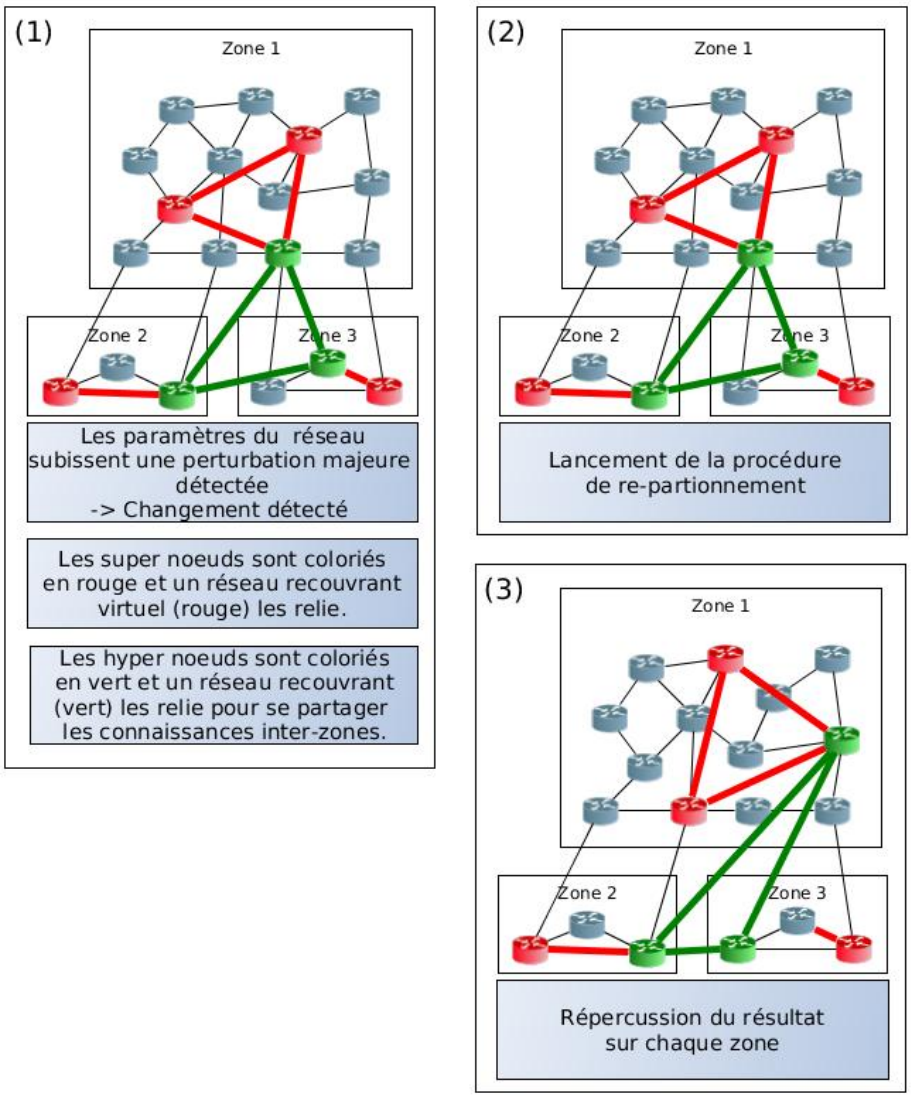


FIGURE 2.4 – Déroulement de l'algorithme adaptatif

2.7.3 Proposition 3 : Granularité du plan de connaissance

Découper le plan de connaissance en plusieurs sous plans est nécessaire à la fois pour assurer un déploiement à large échelle et garantir le cloisonnement de certaines données à des usages limités [Clark et al., 2003]. La finesse de ce découpage, appelée granularité, est un paramètre qui dépend fortement du contexte d'utilisation. Ainsi, Li et al. [2008] proposent une granularité de deux formes :

— Un plan qui gère toutes les connaissances que les agents peuvent rassembler sur le réseau (appelé *Network Knowledge Plane*, NetKP),
— Plusieurs plans pour gérer la connaissance de chaque application (un plan par application : *Specialized Knowledge Planes*, SpecKPs).

De telles méthodes permettent une meilleure prise en charge des aspects liés à la sécurité et au déploiement à large échelle. De plus, elles améliorent les métriques liées à la vitesse de propagation des connaissances (en créant des plans de connaissances réduits impliquant un ensemble plus restreint d'éléments du réseau). La figure 2.5 montre notre vision de la granularité que doit avoir un plan de connaissance. Nous jugeons qu'il est possible de décomposer celui-ci selon la zone de dissémination si les connaissances en question n'ont pas d'impact sur la globalité du réseau. Il est aussi possible de le décomposer selon les priorités données à chaque connaissance afin de les disséminer selon leur degré de priorité. Enfin, il est possible d'opérer selon la nature de l'application cible pour les connaissances à des usages limités.

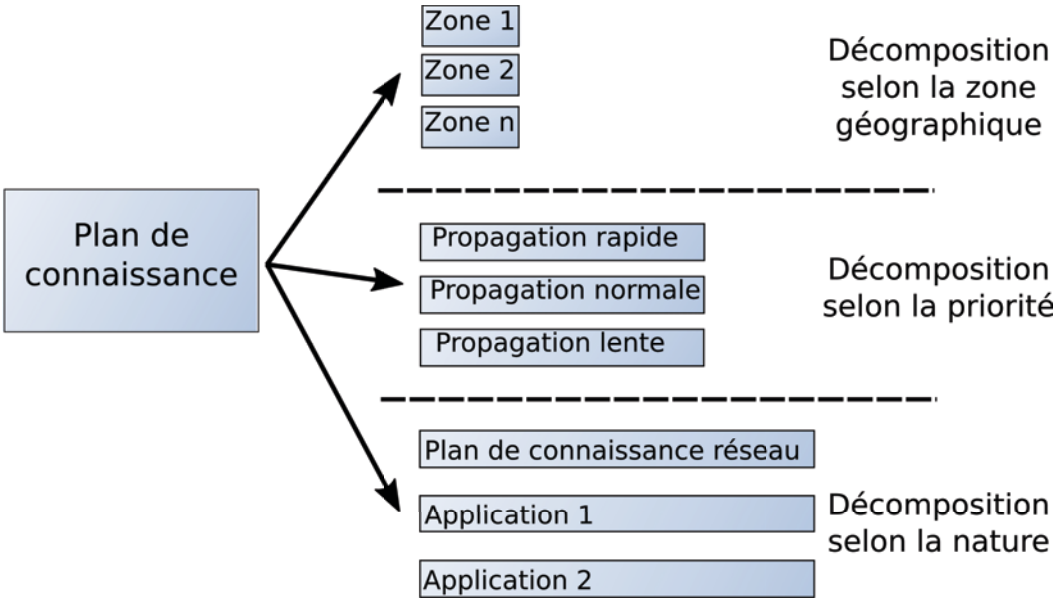


Figure 2.5 – Granularité du plan de connaissance

Partant de l'hypothèse que le transport d'un flux particulier n'est intéressé que par des connaissances particulières, il nous semble préférable de ne diffuser ce type de connaissances que dans le sous-ensemble du réseau intéressé par cette connaissance. Cela permet de confiner les échanges de messages de contôle pour la diffusion et le partage des connaissances uniquement aux zones

considérées. C'est en cela que notre approche consiste à créer et à maintenir des réseaux recouvrants pour le partage des connaissances dépendant de la nature des flux transportés (concept étroitement lié aux applications terminales) comme le montre la figure 2.6. Une fois le choix des noeuds contenant les bases de connaissances effectué, on définit ensuite de manière adaptative et dynamique un réseau recouvrant selon la nature du flux transporté. À titre d'exemple (fig. 2.6), prenons le cas de deux différentes applications transitant simultanément par un réseau opérateur : une application de type audio et une autre de type vidéo. Le réseau virtuel recouvrant construit pour le flux audio est représenté en bleu tandis que celui sur lequel transite les connaissances relatives au transport des flux vidéo est colorié en marron.

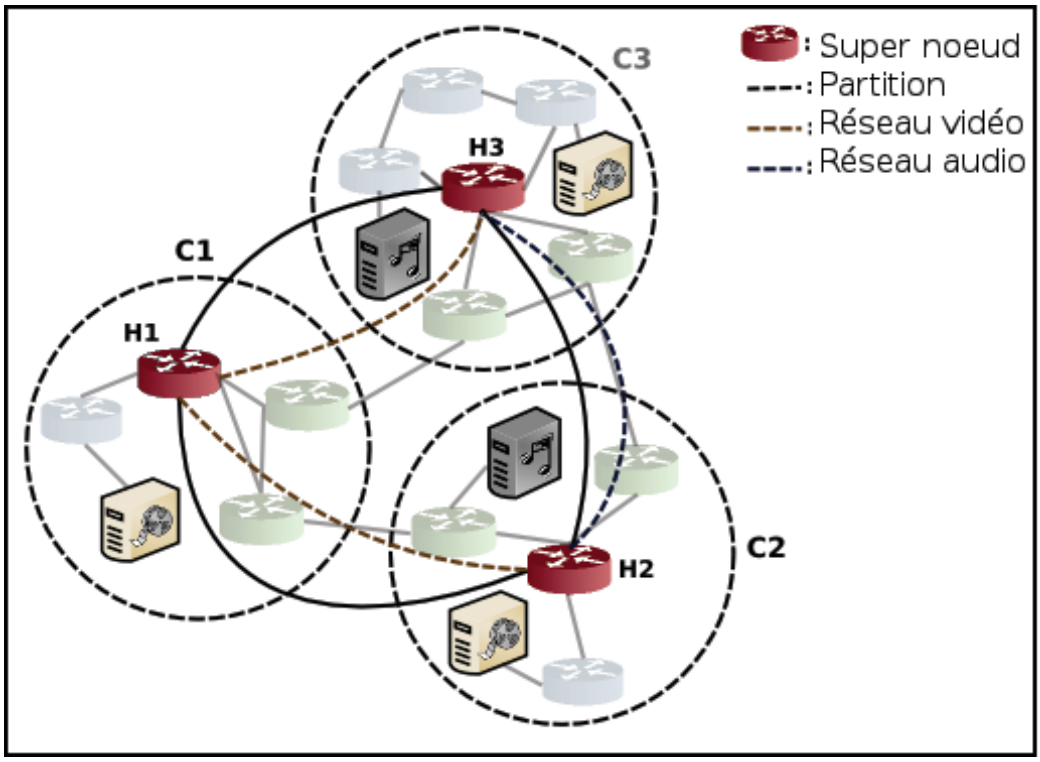


FIGURE 2.6 – Les réseaux de partage des connaissances

De plus, ce découpage par nature d'application peut nous permettre de mixer différents mécanismes de dissémination des connaissances. Par exemple, il devient aisé de fixer les limites d'une "vue située" du moment que le plan de connaissance est divisé en zones. Par ailleurs, la priorité donnée à une connaissance est un facteur relativement important pour améliorer la prise de décision dans un système dans le cas où celui-ci doit traiter l'arrivée de plu-

sieurs connaissances simultanément. Les connaissances de haute priorité (par exemple, rupture d'un lien sur le réseau) devront être disséminées plus rapidement comparativement à des connaissances de basse priorité (par exemple, ajout d'un nouveau contenu dans le catalogue d'un fournisseur de vidéo à la demande) accélérant ainsi une prise de décision adaptée et un faible temps de réaction des éléments de contrôle.

2.8 Conclusion

La propagation de la connaissance sur un ensemble d'équipements répartis dans un réseau représente un des verrous fondamentaux dans la définition du plan de connaissance. Elle doit en effet permettre à la fois un déploiement à large échelle et une adaptation aux changements de différentes sortes impactant le fonctionnement normal du réseau (trafic, topologie, …). Notre solution, proposée dans ce chapitre, se base sur la notion de partitionnement du réseau avec une sélection d'éléments appelés super noeuds.

Dans un premier temps, nous avons formalisé le problème de sélection de ces super noeuds comme un problème de partitionnement K-medoïds et nous avons proposé une nouvelle méthode distribuée pour sa résolution, nommée DCLARA, basée sur l'algorithme CLARA [Kaufman and Rousseeuw, 2008]. Cette proposition consiste à découper le réseau en zones et à sélectionner un noeud (appelé hyper noeud) dans chaque zone. L'ensemble des hyper noeuds se partage les connaissances sur la topologie et les paramètres du réseau afin que chacun d'entre eux puisse dérouler un algorithme de partitionnement (CLARA) et répercuter le résultat obtenu (ensemble des super noeuds sélectionnés) sur sa propre zone. Pour prendre en compte l'aspect dynamique du réseau, nous avons ensuite proposé une amélioration de notre méthode, nommée DCLARA-PH. Cette dernière reconfigure dynamiquement le partitionnement du réseau en utilisant la méthode statistique de détection des changements d'état dans le réseau, dite "test de Page-Hinkley". Elle consiste à évaluer une variable cumulative (somme des délais d'aller-retour entre les noeuds normaux et leur super noeud) afin de détecter, sans aucune hypothèse au préalable, un éventuel changement dans l'architecture réseau sous jacente.

Enfin, dans le but d'améliorer la prise en charge des aspects liés à la sécurité et au déploiement à grande échelle, nous avons proposé une construction du plan de connaissance comme un agrégat d'un ensemble de plans construits en fonction de certains critères fixés a priori par l'utilisateur et pouvant répondre à différents usages : importance de la connaissance, sa position géographique,

type d'application souhaitée, ….. Le plan de connaissance actif sera ainsi instanciée par reconfiguration automatique sur la base de ces plans prédéfinis en amont.

Nous abordons dans le chapitre suivant la conception de la plateforme de gestion de connaissances que nous avons été amenée à développer dans le cadre d'un projet collaboratif de recherche. Nous passons aussi en revue les différentes solutions que nous proposons pour une mise en oeuvre complète d'un plan de connaissance dans le cadre de cette plateforme.

Chapitre 3

Évaluation des approches proposées

Sommaire

3.1 Introduction

Dans ce chapitre, nous abordons, en premier lieu, la conception de notre plateforme de gestion de connaissances. En second lieu, nous évaluons notre algorithme de sélection des super noeuds. Ensuite, nous évaluons, en simulation, notre méthode de dissémination de connaissances par rapport à certaines approches développées récemment ("DHT-Chord" [Stoica et al., 2001], "Diffusion" et "Random Super Nodes" [Abdeljaouad and Karmouch, 2012]). Pour ce faire, nous utilisons trois critères liés à la gestion des connaissances : le temps de diffusion, la surcharge générée (*overhead*) et la latence. Nous étudions aussi un cas de reconfiguration dynamique du réseau dans l'objectif de montrer un usage possible de notre plateforme dans le cadre des réseaux logiciels (*Software Defined Network*, SDN). Enfin, nous validons les performances de notre plateforme expérimentalement en utilisant 2 réseaux recouvrants maillés et un réseau recouvrant à deux niveaux et nous montrons son intérêt dans le cadre d'une application de sécurité.

3.2 Conception de la plateforme

De nombreuses plateformes autonomes ont été développées et comportent des outils et des modules mis à la disposition des administrateurs et des développeurs pour être utilisés et déployés dans un réseau de manière à rendre celui-ci autonome (au sens du concept self-*) (cf. section 1.6.3). Néanmoins, elles ont des visions différentes de la gestion des connaissances (cf. section 1.7.3.4).

Le travail décrit ici s'inscrit dans le cadre du projet Européen [1] CELTIC IPNQSIS (IP Network monitoring for Quality of Service Intelligent Support) [2]. L'objectif de ce projet est de développer des systèmes de surveillance continue des performances du réseau et des services et d'évaluer leur impact sur les utilisateurs finaux. Une innovation importante d'IPNQSIS est l'utilisation, en plus de la Qualité de service (QdS), de la Qualité d'Expérience (QdE) comme élément d'évaluation des services du réseau [Mellouk et al., 2013].

1. Les membres du consortium sont : Université Paris-Est Créteil, Institut Telecom Sud-Paris, IP-Label Newtest, Vierling Communication SAS, EXFO NetHawk, PPO-Yhtiöt Oy, VTT Technical Research Centre, Alcatel-Lucent España SA, Dycec SA, Gigle Semiconductors, Indra Sistemas SA, Naudit SL, Soft Telecom, Acreo AB, Alkit Communications AB, Ericsson AB, Lund University, Procera Networks.
2. http ://ipnqsis.org.

Afin de gérer les informations (paramètres de QdS, QdE, ...) collectées à partir de sondes actives qui permettront de simuler le comportement des utilisateurs dans le cadre des applications qui leur sont adressées, nous avons proposé une nouvelle plateforme de gestion de connaissances. Cette plateforme que nous décrivons ci-dessous, nommée Autolay, est générique et sous licence libre.

3.2.1 Architecture générale

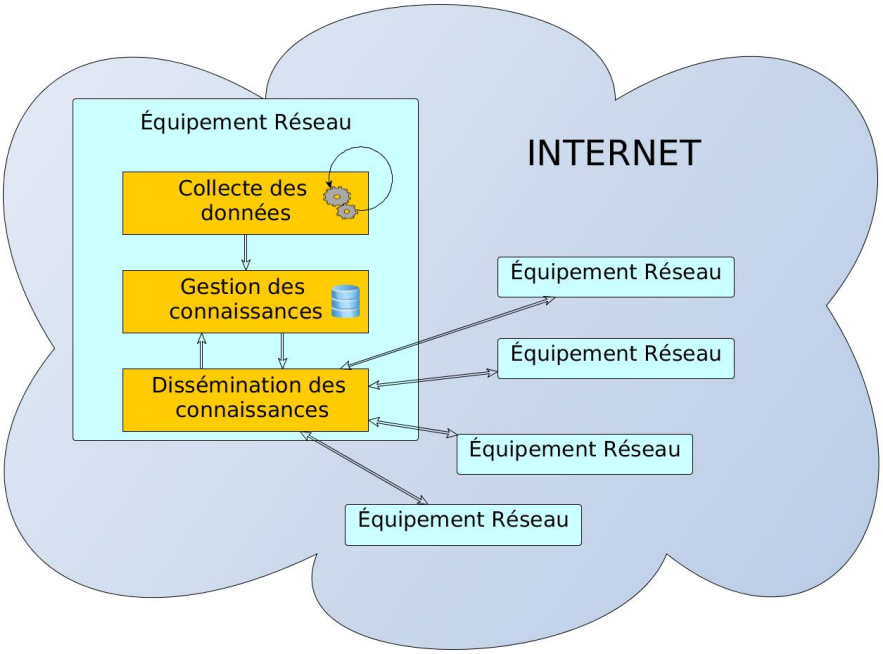


FIGURE 3.1 – Architecture générale de la plateforme de gestion de connaissances.

La plateforme a été conçue sur la base de 3 composants essentiels (cf. fig. 3.1) : un composant chargé de la collecte des données, un autre responsable de la gestion des connaissances (c'est dans ce module que les données et les informations sont transformées en connaissances avant d'être stockées) et un dernier s'occupant de la dissémination des connaissances. Ces trois composants ne sont pas forcément dupliqués dans chacun des éléments du réseau.

Comme l'architecture adoptée pour notre plateforme est hiérarchisée, seuls

les super noeuds (et par extension, les hyper noeuds) sont en charge de la gestion et de la dissémination des connaissances (cf. fig. 3.2). Les noeuds normaux se contentent quant à eux de collecter les données qui seront transformées ensuite en connaissances (cf. section 1.7.3.1).

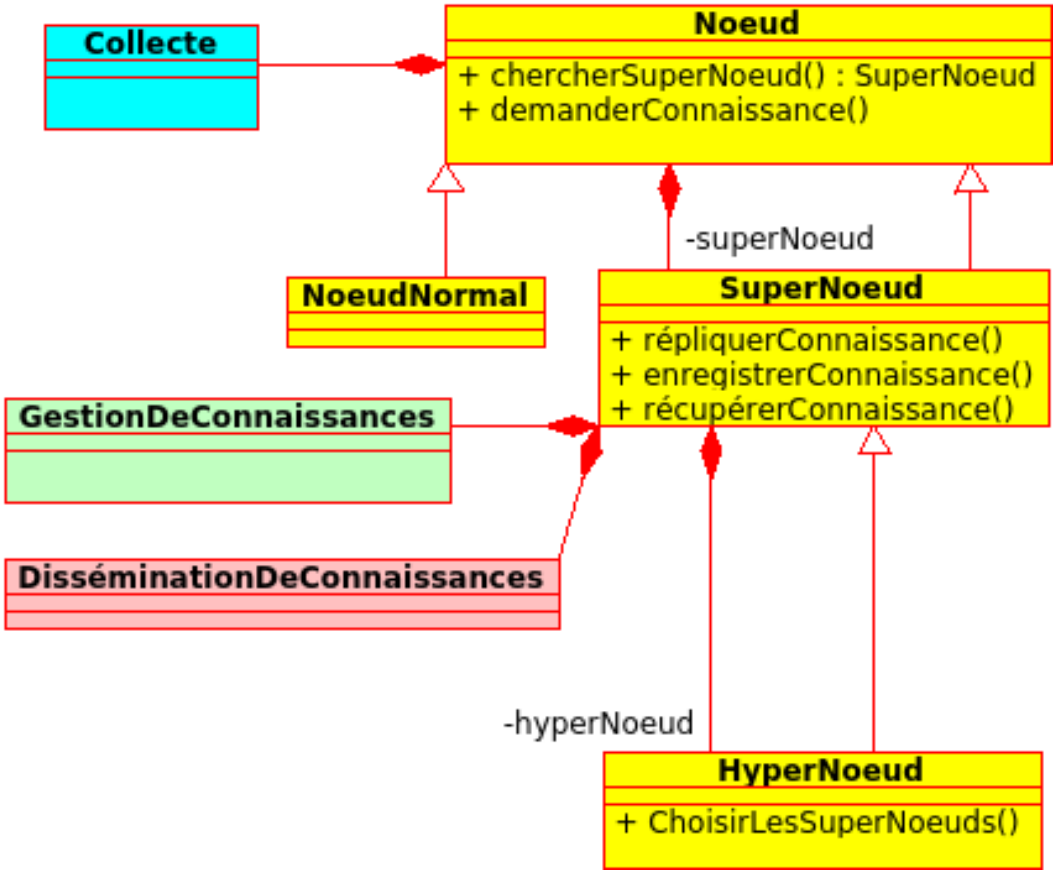


FIGURE 3.2 – Modélisation UML la plateforme de gestion et de dissémination des connaissances.

Nous détaillons dans la suite chacun de ses trois composants.

3.2.2 Collecte des données

La collecte des données constitue la tâche la plus élémentaire pour une plateforme de gestion de connaissances. Il s'agit de monitorer un certain nombre de métriques de performance, en utilisant des capteurs actifs ou des capteurs passifs, dans le but de connaître l'état de chaque élément du réseau à un instant donné. Dans le cadre de ces travaux de thèse, nous avons réalisé un

ensemble de modules de collecte (cf. figure 3.3). Cet ensemble est bien évidemment évolutif en fonction du contexte d'utilisation et des besoins spécifiques de l'application. Nous nous sommes intéressés à trois d'entre eux : le taux de perte, la mesure du délai aller retour et la bande passante résiduelle.

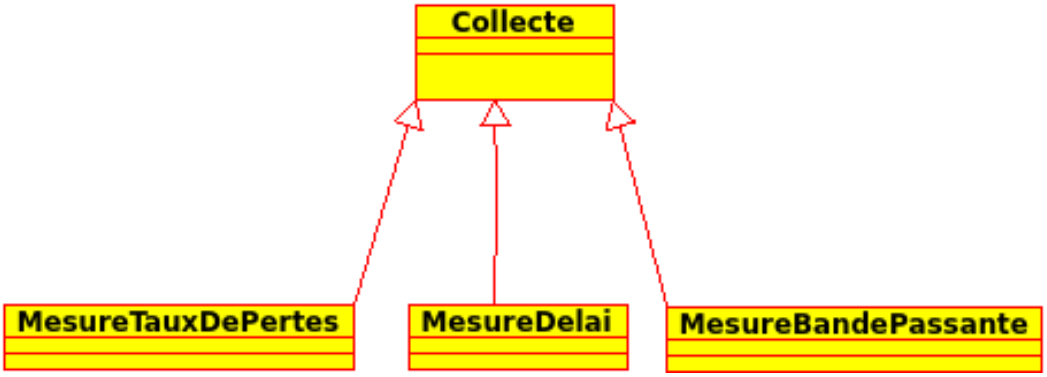


FIGURE 3.3 – Modélisation UML du composant de collecte des données

Ces modules de collecte se basent sur des mesures actives. Les mesures des taux de perte et des délais aller-retour consistent à envoyer des requêtes de type demande/réponse (de type de celles utilisées par le protocole ICMP [3]) et d'en déduire ensuite les valeurs associées à ces deux métriques. La mesure de la bande passante consiste à générer un trafic important entre deux points du réseau (par le biais de l'outil iperf [4]) afin d'évaluer la bande passante résiduelle du lien qui les relie.

Les données récoltées sont ensuite transmises au composant de gestion des connaissances.

3.2.3 Gestion des connaissances

L'une des caractéristiques les plus importantes de notre plateforme est la séparation des connaissances en différentes instances de la base de connaissances. Ainsi, nous pouvons constater, sur la figure 3.4, que la base de connaissance est abstraite. Chaque super noeud peut donc gérer un ensemble de bases de connaissances locales. Ces dernières sont relatives à une zone de propagation ou encore à un type d'application défini a priori. Dans le cadre de cette thèse, nous avons mis en place les trois bases de connaissances suivantes :

3. ICMP : Internet Control Message Protocol.
4. Iperf est développé par le National Laboratory for Applied Network Research

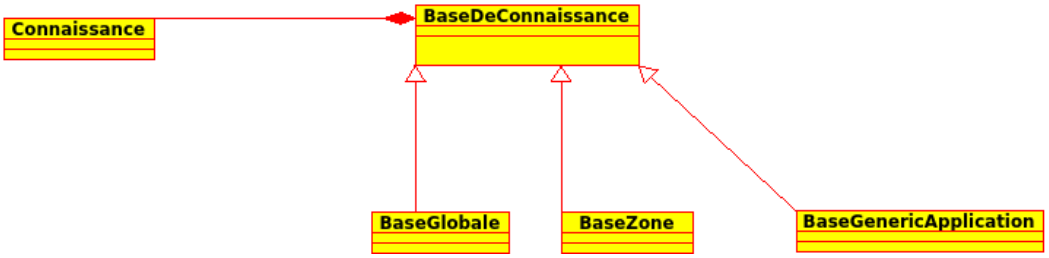


FIGURE 3.4 – Modélisation UML du composant de gestion de connaissances

— Une base globale : utilisée pour gérer les connaissances interdomaines et, plus particulièrement, les connaissances nécessaires au maintien de l'architecture hiérarchique.
— Une base de zone : utilisée pour gérer les connaissances d'une zone du réseau, la portée de la zone restant à prédéfinir. Dans notre cas, il s'agit d'une plage d'adresses IP.
— Une base d'application générique : elle s'occupe de la gestion des connaissances correspondant à une application donnée.

D'un point de vue technique, nos différentes bases sont gérées dans un système de gestion de base de données (SGBD) orienté document de type *Not Only SQL* (NoSQL[5]).

Chaque base de connaissances représente donc une collection de documents de type *JavaScript Object Notation* (JSON[6]). L'intérêt de ce type de SGBD est double :

— Il ne nécessite pas la spécification du schéma des données correspondant. Prenons l'exemple de la figure 3.5, le document n° 1 (correspondant au noeud 1) a une structure différente du document numéroté 2 (correspondant au noeud 2). Ils se différencient en effet par leur structure atomique ("number of interfaces" dans un cas et "service" dans l'autre).
— Toutes les versions d'un document peuvent être sauvegardées permetant ainsi de récupérer facilement des informations sur l'état antérieur du réseau.

(NLANR) : http ://www.nlanr.net/.

5. NoSQL désigne une catégorie de systèmes de gestion de base de données (SGBD) qui n'est plus fondée sur l'architecture classique des bases relationnelles.

6. JSON est un format de données textuel et générique

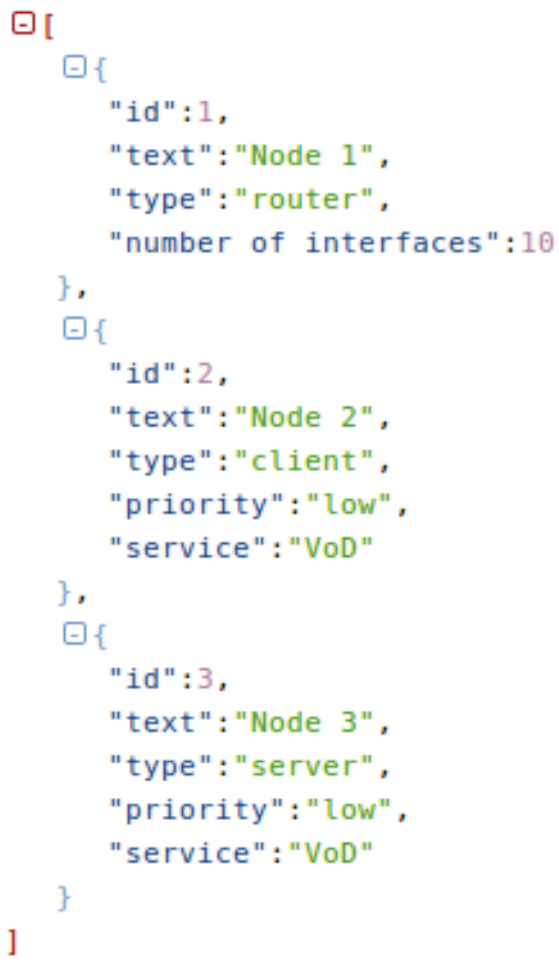


FIGURE 3.5 – Exemple d'une collection de connaissances.

Citons au passage qu'un document est défini par un identifiant unique et un numéro de version nécessaire pour son partage entre les noeuds.

3.2.4 Dissémination des connaissances

La finalité de la plateforme que nous proposons est de pouvoir disséminer la connaissance dans le réseau de la manière la plus optimisée possible. Pour y arriver, les documents (qui représentent chacun une connaissance donnée) sont classés selon leur ordre de priorité (voir figure 3.6). Ceux-ci sont classées en 3 catégories : haute priorité, moyenne priorité et basse priorité. Les règles qui fixent les priorités pour chaque connaissance sont, pour l'instant, fixées par l'utilisateur de la plateforme. À terme, ceci peut être étendu, par la suite,

par le développement de modules permettant de fixer ces priorités sur la base de règles d'apprentissage.

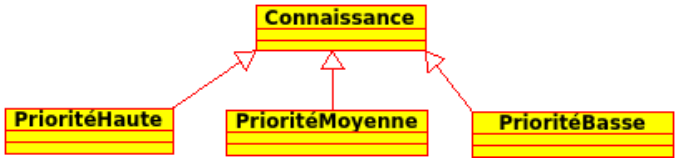


FIGURE 3.6 – Modélisation UML des connaissances.

Pour chaque type de connaissances de toute priorité est instancié un réseau virtuel recouvrant spécifique reliant un ensemble de super noeuds et d'hyper noeuds du réseau (fig 3.2). Deux sortes de réseaux recouvrants ont été mis en oeuvre : des réseaux maillés et des réseaux à 2 niveaux hiérarchiques.

Dans les réseaux maillés, les super noeuds (ou les hyper noeuds) sont tous reliés les uns aux autres comme le montre la figure 3.7. Les connaissances sont transmises d'un super noeud vers ses k plus proches voisins en évitant, si possible, de les transmettre à des noeuds qui en disposent déjà (k est fixé par l'utilisateur de la plateforme selon ses besoins). Ces derniers les répliquent à leur tour de la même manière. Ce type de réseau recouvrant est robuste dans la mesure où il n'y a aucun point de concentration.

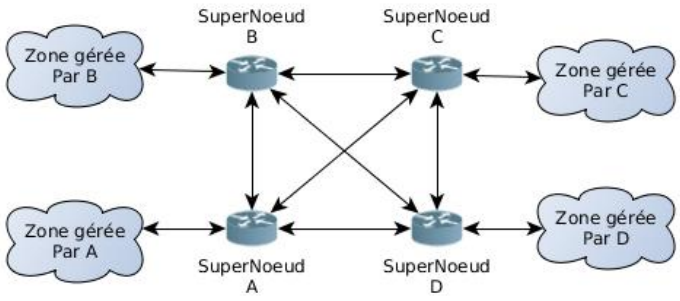


FIGURE 3.7 – Architecture d'un réseau maillé

Dans les réseaux recouvrants à 2 niveaux hiérarchiques, les super noeuds sont reliés à un hyper noeud comme le montre la figure 3.8. Les connaissances sont disséminées d'un super noeud vers l'hyper noeud qui les réplique ensuite dans les autres super noeuds. Ce type de réseau recouvrant est surtout utilisé pour la dissémination des connaissances liées à la topologie du réseau en raison de la convergence rapide de la base de connaissances au niveau de l'hyper noeud (qui est à la charge du processus de maintien de la structure à base de super noeuds).

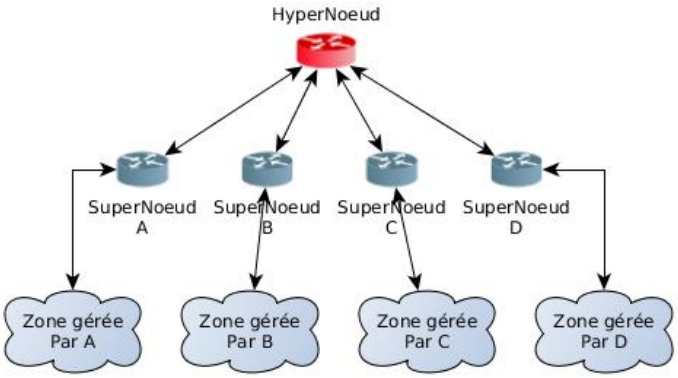


FIGURE 3.8 – Architecture d'un réseau à deux niveaux

Dans le cas d'un réseau étendu, une topologie mixte peut être envisagée : les super noeuds d'une zone sont reliés à un hyper noeud et ces derniers sont eux mêmes reliés entre eux au travers d'un réseau maillé (voir figure 3.9).

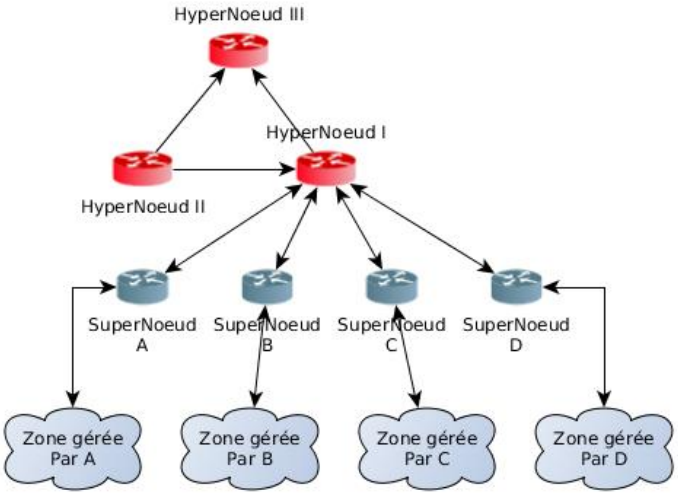


FIGURE 3.9 – Architecture d'un réseau mixte

3.3 Validation de l'algorithme de sélection des super noeuds

Afin de comparer l'algorithme D-CLARA proposé (décrit dans le chapitre 2) pour la construction de la topologie reliant les super noeuds entre eux,

nous avons considéré un algorithme de la classe centralisée. Pour ce faire, nous avons pris en compte l'algorithme SPSP [Wolf and Merz, 2007]. Dans la littérature, ce dernier est fourni avec un ensemble de données de comparaison que nous décrivons dans la suite de ce document. Sur la base de ces données, nous avons mis en place un simulateur pour évaluer notre approche. Dans ce simulateur, nous considérons que chaque noeud connaît uniquement les délais aller-retour avec son voisinage.

3.3.1 Description des données

Les données utilisées par l'algorithme SPSP sont fournies sous la forme de matrices symétriques $n - \times - n$ qui représentent les distances entre différents noeuds du réseau. Ces distances sont exprimées en millisecondes (ms) et correspondent aux valeurs moyennes arrondies de la mesure du temps aller-retour (*Round Trip Time*, RTT). Ces valeurs sont issues de mesures réelles obtenues dans le cadre du réseau PlanetLab [7] [Culler et al., 2002].

Le tableau 3.1 représente la description de ces données.

Instance	nbre de noeuds	nbre de supernoeuds	capacité d'un supernoeud
01	127	12	22
02	321	19	34
03	324	18	36
04	70	9	16
05	374	20	38
06	365	20	38
07	380	20	38
08	402	21	40
09	419	21	40
10	414	21	40
11	407	21	40
12	414	21	40

TABLE 3.1 – Déscription des données fournies par SPSP

Le tableau 3.1 est composé de 12 lignes où chaque ligne correspond aux données récoltées sur un mois. Pour chaque instance, il fournit le nombre des

7. PlanetLab est un réseau mondial de recherche qui soutient le développement de nouveaux services de réseau créé en 2003. Il est actuellement composé de 1170 noeuds répartis sur 561 sites.

noeuds du réseau, le nombre de super noeuds souhaités ainsi que la capacité maximum souhaitée (en terme de nombre de noeuds qu'il peut gérer) pour chaque super noeud.

3.3.2 Résultats de la simulation

L'évaluation des deux algorithmes (D-CLARA et SPSP) se fait sur la base de la métrique de dissimilarité totale (la somme des délais aller-retour entre l'ensemble des noeuds du réseau et leur super noeud respectif). Cette dissimilarité impacte la latence lors de l'envoi d'une requête pour la récupération d'une connaissance. Les résultats de simulation sont résumés dans la figure 3.10.

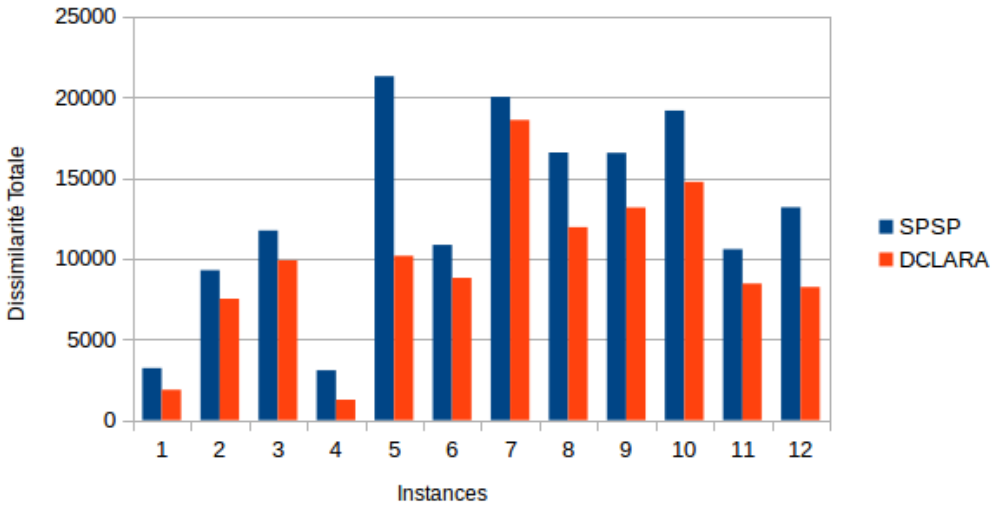


FIGURE 3.10 – Compraison des algorithmes de sélection des super noeuds

Il apparaît clairement que, pour chacune des 12 instances étudiées, notre proposition (D-CLARA) offre de meilleurs résultats comparativement à ceux obtenus par la méthode SPSP en termes de dissimilarité totale. Dans certains cas, la différence entre les dissimilarités obtenues avec les deux approches est du simple au double. Par exemple, pour l'instance numéro 5, D-Clara obtient une dissimilarité de 21293ms contre 10173ms pour SPSP. Il est à noter que plus la valeur de la dissimilarité totale est grande, plus le temps requis pour récupérer une connaissance est grand, impactant, négativement, le temps de réaction d'un noeud à l'évènement constaté.

3.4 Évaluation de la plateforme de gestion des connaissances en simulation

Dans cette section, nous décrivons la plateforme de simulation et les approches utilisées dans le cadre de cette étude (DHT-chord, diffusion, random super nodes, DClara et DClara-PH). Nous considérons trois critères de comparaison :

— Le temps nécessaire pour diffuser les connaissances à l'ensemble des noeuds du réseau,
— Le temps de réponse à une requête (la latence),
— La surcharge générée par chaque noeud.

Le temps de diffusion impacte la cohérence des connaissances et l'adéquation des réactions dans le réseau. Le temps de réponse à une requête informe sur la capacité de l'ensemble du réseau à réagir rapidement et à adapter en conséquence sa décision. La surcharge (*overhead*) liée au flux de contrôle induit par les approches utilisées impacte les performances du système (occupation de la bande passante par ce flux de contrôle).

3.4.1 Plateforme de simulation

Dans cette phase, nous avons opté pour le simulateur Omnet++ [Varga and Hornig, 2008]. Il s'agit d'un simulateur d'évènements basé sur le langage C++, destiné principalement à simuler les protocoles réseau et les systèmes distribués. Il est totalement programmable, paramétrable et modulaire. C'est une application en code source libre sous licence GNU, développée par Andras Varga, chercheur à l'université de Budapest.

Le fonctionnement d'Omnet++ repose entièrement sur l'utilisation de modules qui communiquent entre eux par le biais de messages. Ces modules sont organisés hiérarchiquement. Les modules de base sont appelés "les modules simples". Ils sont regroupés en modules composés qui peuvent eux-mêmes être regroupés en d'autres modules composés. Le nombre des niveaux hiérarchiques n'est pas limité.

Les modules sont codés en C++ et sont des instances du type module de base. L'architecture est construite de telle sorte que "les modules simples" sont à la fois les émetteurs et les destinataires des messages. Les modules composés se contentent de relayer les messages aux différents modules simples

de façon transparente. On peut attribuer différents paramètres aux connexions reliant les modules : des délais de propagation, des débits de données, des taux d'erreurs, …

Les messages sont transmis par le biais de portes (*gates*). Les portes représentent les interfaces d'émission et de réception des modules et une connexion relie une porte d'entrée à une porte de sortie. une connexion n'est créée que dans un seul niveau de hiérarchie des modules. Il est par exemple impossible de créer une connexion directe entre un module et un sous-module d'un autre module de même niveau dans la hiérarchie.

Un exemple de ce type d'architecture serait deux réseaux locaux (LANs) reliés par un routeur, chaque poste ou élément de chacun des réseaux pourrait être représenté par un module simple, alors que chacun des LANs constituerait un module composé. Ces deux modules composés seraient reliés par un module simple qui représenterait un routeur.

Il existe divers composants logiciels structurels que l'on peut ajouter au simulateur Omnet++ afin de supporter des fonctionnalités additionnelles. Dans notre cas, nous nous sommes intéressés aux composants supportant des réseaux recouvrants (overlay). OverSim [Baumgart et al., 2007] semble être le composant le plus abouti. Celui-ci est en code source libre et cible la simulation des réseaux overlay et des réseaux peer-to-peer. Il contient plusieurs modèles structurés (ex. Chord, Kademlia, Pastry) et non structurés (ex. GIA) de systèmes P2P et de réseaux overlay. Cependant, OverSim, aussi abouti qu'il puisse être, s'intéresse en grande partie aux réseaux overlay de niveau applicatif. De ce fait, il ne convient pas à notre approche. Nous avons donc développé et implémenté une amélioration de ce composant sur la base de la plateforme décrite dans la section 3.2.

3.4.2 Toplogie du réseau

La topologie de réseau utilisée est celle basée sur la topologie de Waxman [Waxman, 1991]. En utilisant cette méthode basée sur la distribution statistique, les noeuds sont uniformément déployés dans le réseau. Les liens sont ajoutés selon la probabilité définie par l'équation 3.1. Cette probabilité dépend de la distance entre les noeuds.

$$P = \alpha \times exp(-d/L \times \beta) \tag{3.1}$$

Dans l'équation 3.1, d est la distance entre chaque paire de noeuds dans le réseau, L est la distance maximale dans le réseau, $\alpha > 0$ et $\beta <= 1$. Ces paramètres fixent la probabilité d'une connexion (la présence d'un lien entre deux noeuds) et β contrôle explicitement le rapport entre les liens longs et les liens courts.

3.4.3 Scénario de simulation

Cinq méthodes de diffusion sont évaluées dans la suite :

— Un algorithme de DHT-Chord [Stoica et al., 2001] (noté DHT)
— Un algorithme de diffusion de type glouton (chaque noeud envoie périodiquement des informations à tous les autres noeuds) (noté Broadcast)
— Une approche hiérarchique basée sur une sélection aléatoire de super noeuds [Abdeljaouad and Karmouch, 2012] (notée Random)
— Notre première contribution basée sur une approche hiérarchique (notée DCLARA)
— Notre seconde contribution basée sur une approche adaptative utilisant le test de Page-Hinkley (notée DCLARA-PH).

Les paramètres de simulation sont décrits dans le tableau 3.2.

Paramètres	Valeurs
Nombre de noeuds dans le réseau	[50 → 5000]
α	0.01
β	0.08
Nombre de voisins maximum	5 noeuds
Capacité des super noeuds	10 noeuds
λ	100ms
σ	+10ms
Intervalle entre deux sondes	250ms
Taille d'un probe	64kbits
Intervalle entre deux paquets de connaisance	500ms
Taille d'un paquet de connaisance	[100-500 Kbits]
Nombre de miroir pour le DHT	3

TABLE 3.2 – Paramètres de simulation

Un réseau d'une capacité de 200 à 5000 noeuds a été généré à des fins de comparaison. La topologie générée ainsi est modifiée de manière à simuler le caractère dynamique d'un véritable réseau (simuler des pannes de liens et de

noeuds dans le réseau). Nous considérons que chaque super noeud peut être en charge d'un nombre fini de noeuds. Dans la pratique, la capacité d'un super noeud dépend à la fois de ses ressources et des besoins de l'application. Dans notre exemple, nous avons choisi, arbitrairement, de la fixer à 10 noeuds.

Deux messages représentant des connaissances sont générés chaque seconde et chaque noeud du réseau envoie 4 sondes à ses voisins toutes les secondes pour évaluer en permanence la qualité des liens du réseau. Le nombre de miroirs pour le DHT est fixé à 3 et le paramètre du test de Page-Hinkley à 100ms avec une tolérance de 10 ms.

3.4.4 Résultats de la simulation

Cette section détaille les performances obtenues des algorithmes étudiés en fonction des trois critères décrits précédemment : le temps de la diffusion, la surcharge du système (overhead) et le temps de réponse à une requête (latence).

3.4.4.1 Delai de convergence d'une connaissance

Il est communément admis que le facteur le plus important pour évaluer un mécanisme de diffusion de connaissances consiste dans le temps nécessaire pour la propagation d'une connaissance à tous les noeuds du réseau. En effet, plus ce temps est court, plus la décision prise par les applications de contrôle est appropriée et prend en compte une information réelle qui représente fidèlement le réseau. Ce paramètre est appelé "délai de convergence". La figure 3.11 illustre, pour chaque algorithme testé, ce paramètre qui est donné en millisecondes.

La figure 3.11 montre que le mécanisme de diffusion est celui qui permet d'assurer la cohérence des connaissances dans le réseau. En fait, chaque noeud envoie les nouvelles connaissances apprises à tous les autres noeuds du réseau chaque 500ms. L'approche DHT estime que l'information est diffusée une fois qu'elle est reproduite à n noeuds dans le réseau. Les performances des trois autres approches hiérarchiques dépendent du mécanisme de sélection des super noeuds : Random est le plus lent tandis que DCLARA-PH offre les meilleurs délais. En effet, DCLARA-PH nécessite approximativement 1 seconde pour diffuser des connaissances dans toutes les parties du réseau.

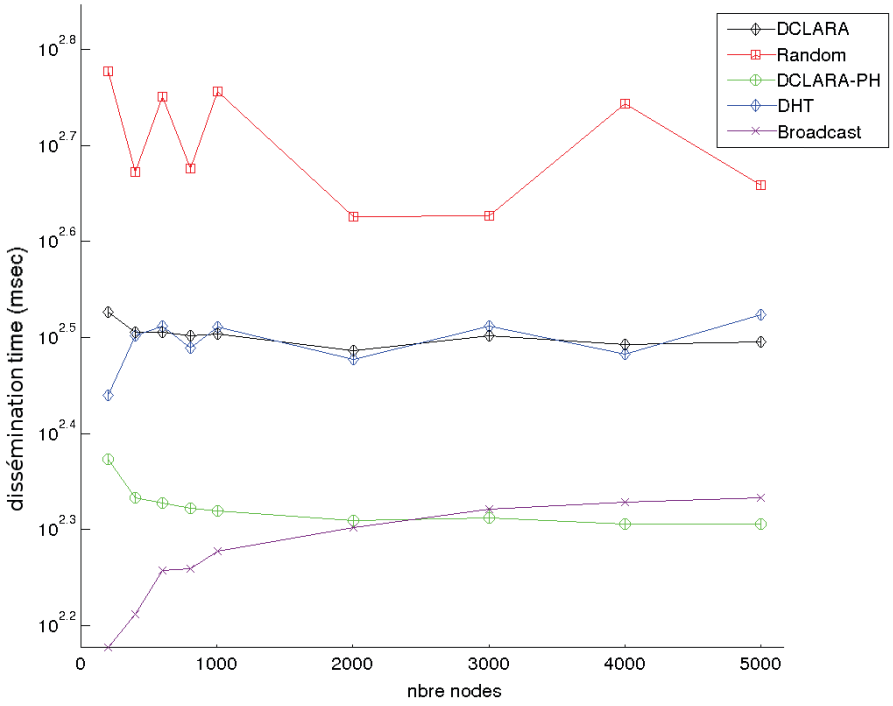


FIGURE 3.11 – Delai de convergence dans les approches testées

3.4.4.2 Surcharge générée

La surcharge relative du système peut être estimée à partir des flux de contrôle induit par chaque algorithme de diffusion des connaissances. Une surcharge très élevée peut conduire à une saturation des liens dans le réseau. Les paquets de contrôle (liés au maintien de la topologie de dissémination) et de connaissances peuvent mener à une congestion potentielle du système. La figure 3.12 montre la surcharge provoquée par chaque méthode de diffusion des connaissances (en kbits/s).

Ces résultats montrent que le mécanisme de diffusion n'est pas possible dans un scénario réel. En effet, il représente 1000 fois la valeur obtenue par les autres approches. Le surcoût généré par les approches hiérarchiques ainsi que par DHT demeure constant indépendamment du nombre de noeuds dans le réseau. Toutefois, les approches hiérarchiques (DCLARA, DCLARA-PH et Random) sont 1,5 fois mieux que l'approche DHT et valident de la sorte l'intérêt de nos approches.

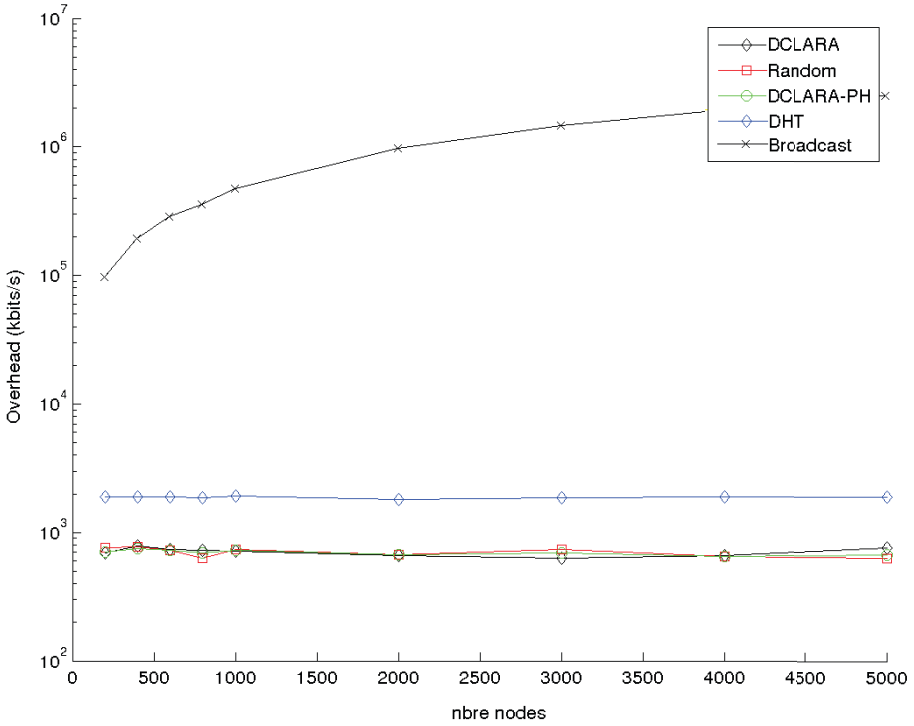


FIGURE 3.12 – Overhead des approches testées

3.4.4.3 Latence lors de la récupération d'une connaissance

Étant donné la forte dynamicité du système, les connaissances peuvent déprécier rapidement et devenir ainsi obsolètes. C'est pourquoi le temps de recherche est un paramètre crucial dans les systèmes temps réel. En effet, une latence très élevée peut conduire à des décisions inappropriées. La figure 3.13 résume les résultats obtenus dans le calcul de la latence (en ms).

L'approche de diffusion est une méthode où la connaissance est disponible localement pour chaque noeud du réseau, ce qui explique la latence nulle obtenue. Dans les approches hiérarchiques, chaque noeud peut récupérer une connaissance à partir de son super noeud. Le temps de recherche est toujours constant indépendamment du nombre de noeuds dans le réseau dans ce type d'approches. DCLARA-PH est l'approche hiérarchique qui offre les meilleurs résultats. Elle optimise, en effet, en permanence la topologie de diffusion. Dans DHT, la latence est logarithmique. L'approche DHT est inappropriée dans notre cas. En fait, la latence dans cette approche augmente avec le nombre de

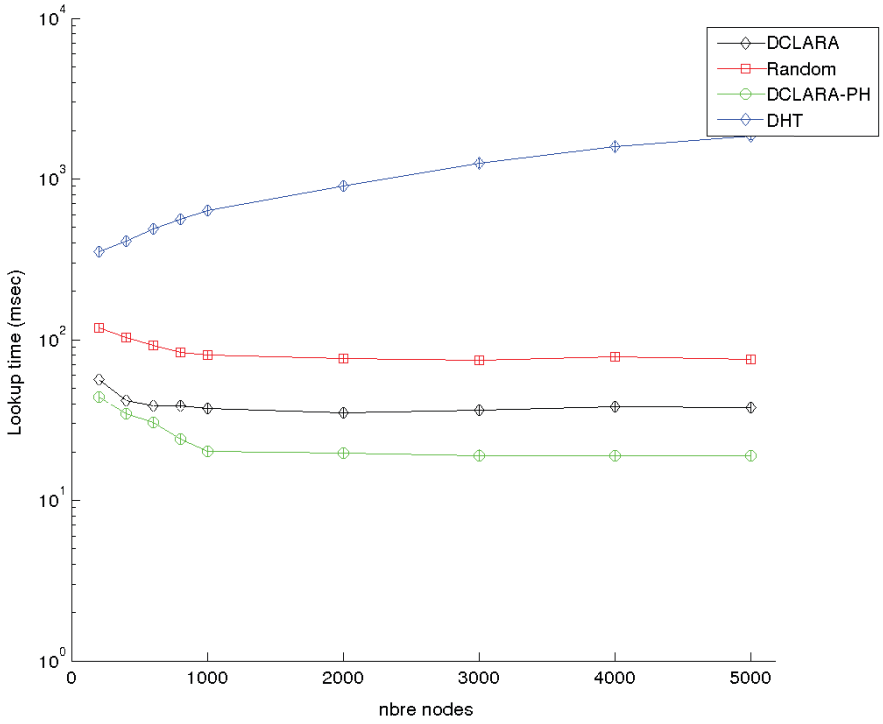


FIGURE 3.13 – Latence des approches testées

noeuds et dépasse de 100 fois celle obtenue par l'algorithme DCLARA-PH.

3.4.5 Cas applicatif : Reconfiguration dynamique des réseaux de distribution de contenu

Dans le but de montrer l'intérêt de la plateforme développée dans un contexte de réseau logiciel (*Software Defined Networks*, SDN), nous simulons un scénario de reconfiguration dynamique des réseaux de contenu. La simulation consiste à configurer deux réseaux recouvrants : le premier est en charge de la distribution des flux audio tandis que le second est construit pour la distribution des flux vidéo comme le montre la figure 3.14. Les flux audio (représentés par un trafic à débit constant (*Constant Bit Rate*, CBR) transitent entre le serveur SA et le client CA utilisant le chemin noté "audio 1" tandis que les flux vidéos (représentés par un trafic de type Poissonnien) circulent entre le serveur SV et le client CV en utilisant le chemin appelé "vidéo".

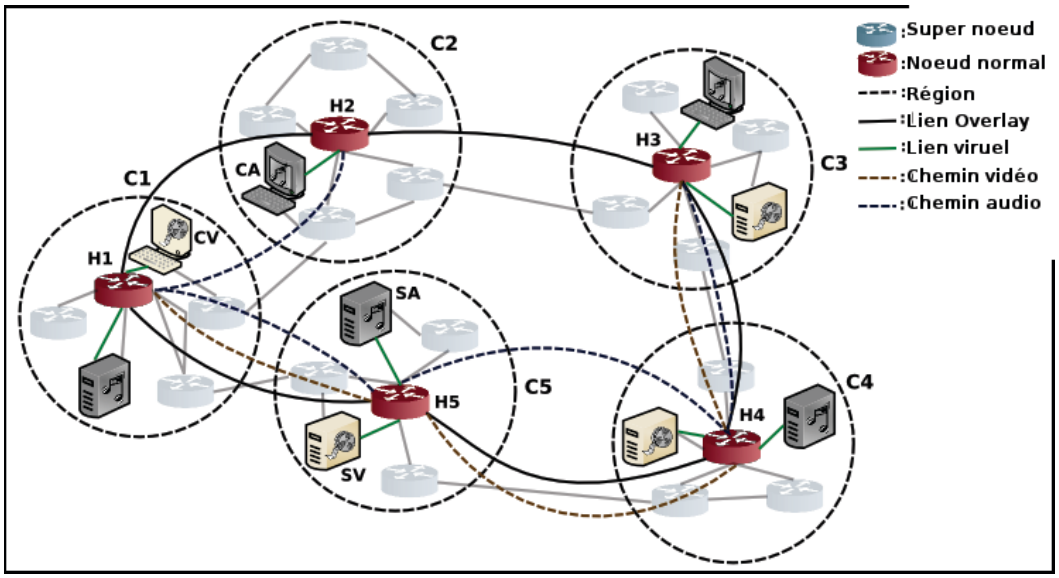


FIGURE 3.14 – Configuration initiale

Nous augmentons successivement l'intensité des flux vidéo transitant par le chemin "vidéo" afin de perturber les flux audio. À partir d'un certain seuil de taux d'erreur, le trafic audio est considéré comme non satisfaisant et il est rerouté par un autre chemin offrant de meilleures performances. Notons au passage ici que la notion de chemin n'est pas celle qu'on retrouve classiquement dans le routage, mais il s'agit ici du réseau virtuel recouvrant. Notre approche consiste à reconfigurer dynamiquement ces réseaux virtuels recouvrants sur la base des informations collectées dans la base de connaissances.

Le seuil à partir duquel la plateforme décide de rerouter un flux est difficile à définir. Dans notre cas, nous nous sommes basés sur la perception de l'usager relative à la notion de qualité d'expérience (QdE) [Fiedler et al., 2010]. Nous avons donc au préalable récolté expérimentalement des mesures de QdE en fonction du taux d'erreur d'un flux audio comme le montre le tableau 3.3. La QdE est exprimée ici en MOS (Mean Opinion Score)[Rec, 2006]. Il s'agit d'une note donnée par l'utilisateur pour caractériser la qualité de la restitution d'un flux. La note peut varier entre 1 (très mauvais) et 5 (excellent, comparable à la version d'origine). Une note inférieure à 3 est donc considérée comme non satisfaisante et le réseau doit trouver une autre route améliorant les performances du rendu final. Autrement dit, si le super noeud H5 auquel est relié le serveur audio se rend compte que la qualité perçue par l'usager (QdE) n'est plus satisfaisante (la QdE étant une des connaissances partagées à travers notre plan de connaissance), il engage une procédure de reconfiguration du

réseau recouvrant relatif au flux audio.

Taux d'erreur	**Score MOS (Mean Opinion Score)**
0-1,5	5
1,5-4	4.5
4-15	4
15-20	3
20-30	2
30-50	1.5
plus de 50	1

TABLE 3.3 – Perception de l'usager en fonction du taux d'erreur d'un flux audio

Les paramètres de simulation des flux audio et des flux vidéo sont représentés dans le tableau 3.4. Nous fixons la taille de toutes les files d'attente à 10 paquets et le débit de tous les liens à 20Mbps.

Paramètres	**Valeur**
Type du flux audio	CBR
Taille des paquets audio	100 bits
Débit du flux audio	10 Mbits
Type du flux vidéo	Source On/Off
Taille des paquets vidéo	1440 bits
Temps de repos de la source On/Off	200 ms
Temps de burst de la source On/Off	100 ms

TABLE 3.4 – Paramètres de simulation

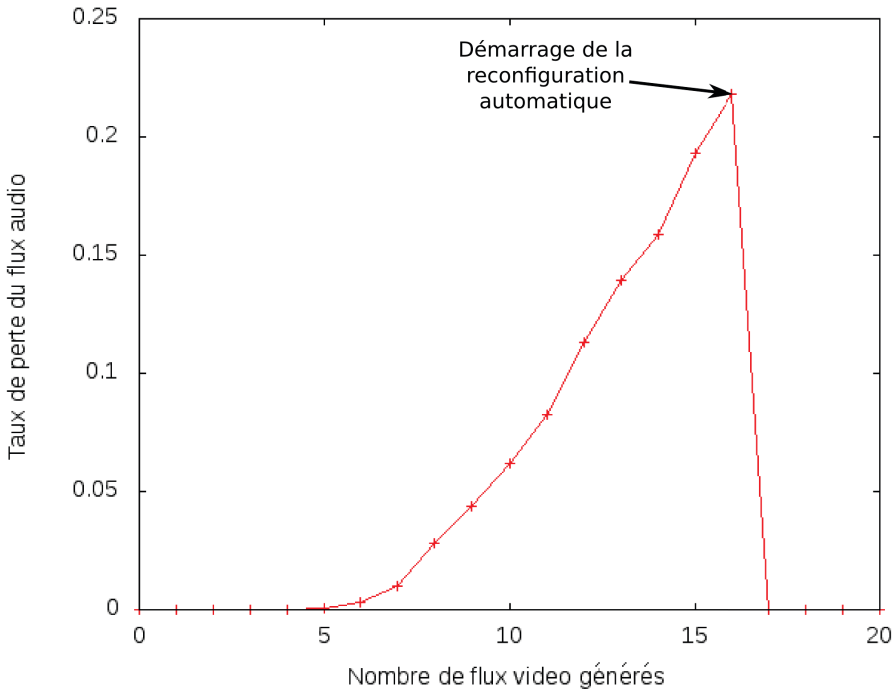


FIGURE 3.15 – Taux d'erreur du flux audio en fonction de la densité du trafic issu du flux vidéo

Nous augmentons progressivement le nombre de paquets issus des flux vidéo de manière à saturer la bande passante. Une fois le taux d'erreur des flux audio dépassant la valeur fixée de 20% (considérée comme mauvaise au regard de la QdE collectée (cf. figure 3.15)), une reconfiguration de l'arbre de dissémination du réseau recouvrant dédié au flux audio est enclenchée afin de lui assurer un meilleur rendu final, comme illustré dans la figure 3.16.

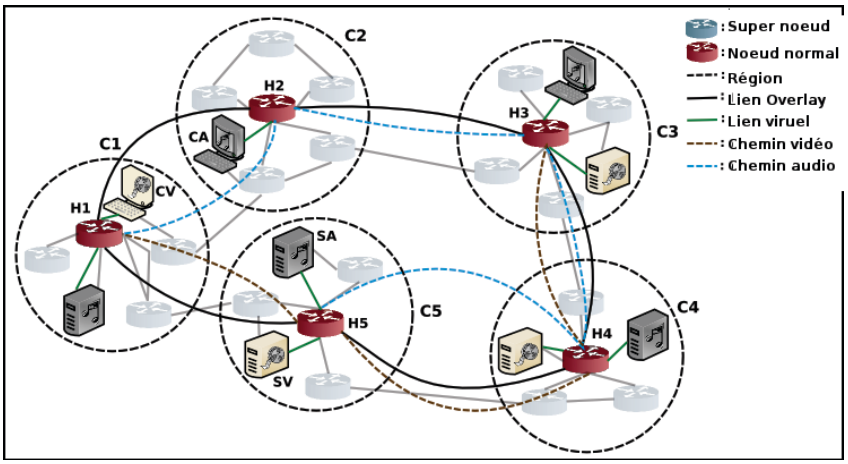


FIGURE 3.16 – Configuration finale

3.5 Évaluation expérimentale de la plateforme de gestion de connaissances

Après avoir testé notre plateforme de gestion de connaissances sur un simulateur, nous avons effectué une validation sur un cas réel. Pour ce faire, nous avons développé la plateforme d'expérimentation NExTLab (voir annexe A) qui permet de créer une topologie de réseaux virtuels.

Les expérimentations testées ont pour but d'étudier la sélection des réseaux recouvrants les plus adéquats pour la dissémination des 3 types de connaissances considérées : haute priorité, moyenne priorité et basse priorité. Pour chacun de ces types, le réseau recouvrant correspondant est instancié.

Afin de montrer l'intérêt de l'utilisation de notre plateforme dans un contexte des réseaux logiciels, nous proposons de l'étudier dans un scénario défini par une attaque de type déni de service distribué (*Distributed Denial of Service*, DDoS) sur l'ensemble du réseau.

3.5.1 Topologie du réseau considéré

Pour l'évaluation expérimentale de la plateforme, nous avons reproduit sur NExTLab la topologie du réseau backbone NTT (Nippon Telephone and Telegraph Company) [Rubio-Largo et al., 2010b] (voir figure 3.17) qui sert souvent de réseau exemple dans la validation des propositions au sein de la

communauté scientifique [Rubio-Largo et al., 2010a ; Roa et al., 2009 ; Tran et al., 2013]. Ce réseau est constitué de 55 noeuds et de 74 liens bidirectionnels.

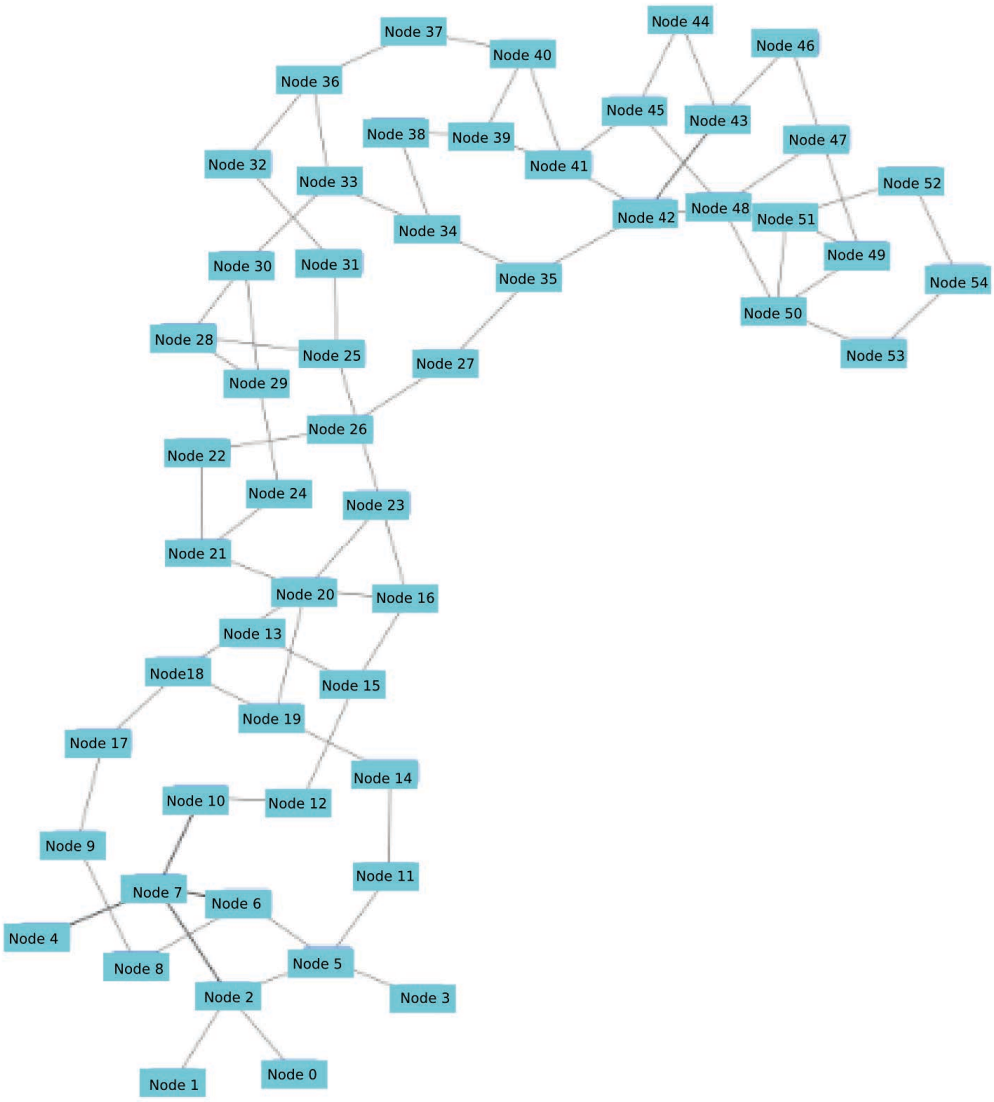


FIGURE 3.17 – Topologie du réseau NTT

3.5.2 Construction de la topologie de gestion des connaissances

Afin de sélectionner et de maintenir les noeuds en charge de la gestion des connaissances, nous avons déployé des agents intelligents sur l'ensemble des

noeuds du réseau qui utilisent notre proposition DCLARA-PH (décrit dans le chapitre 2, section 2.7.2). Dans la figure 3.18, les super noeuds sélectionnés sont entourés en rouge et l'hyper noeud en vert. Les traits interrompus correspondent à la délimitation des zones gérées par les super noeuds.

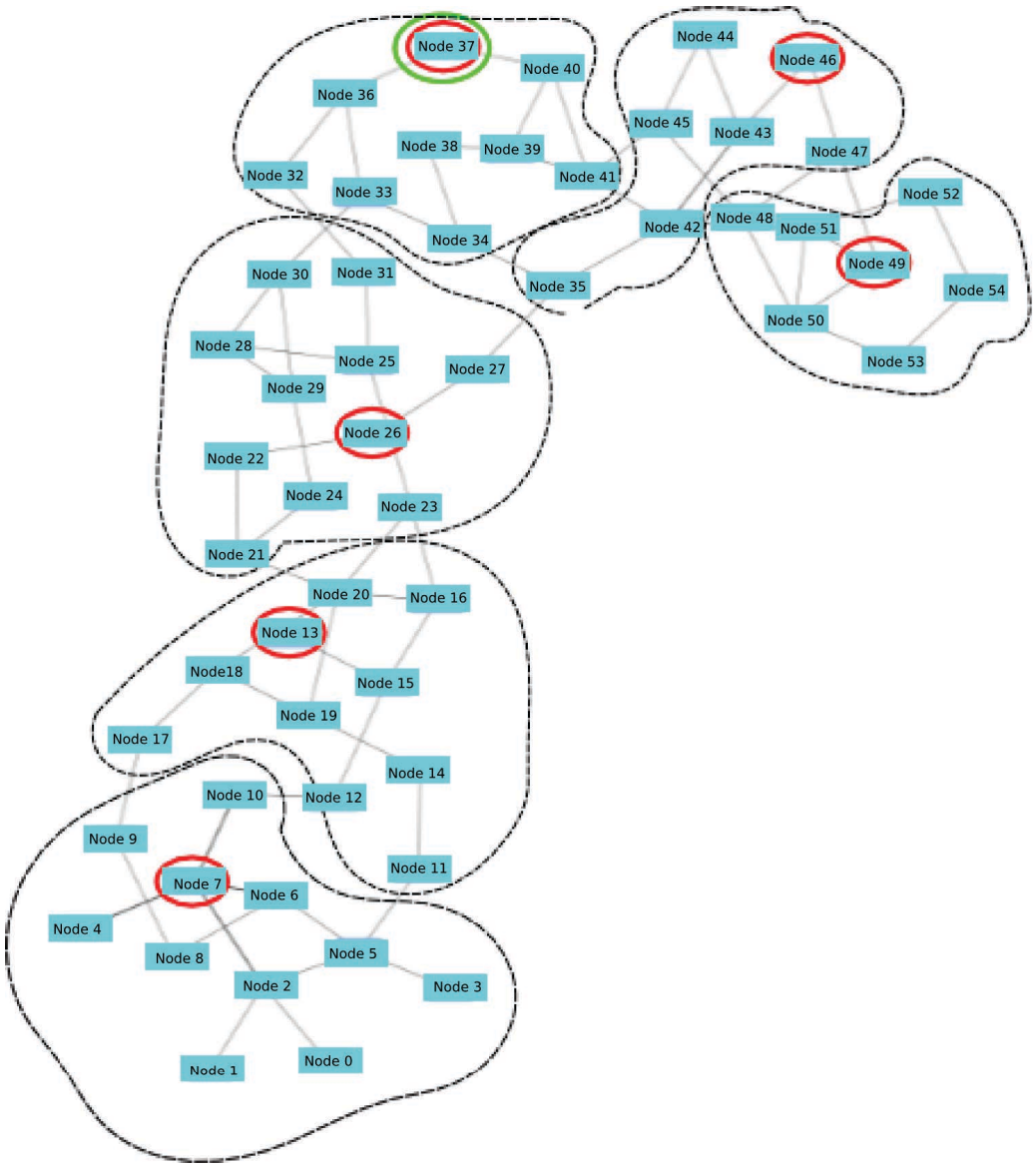


FIGURE 3.18 – Sélection des super noeuds et des hyper noeuds

Les noeuds sélectionnés sont reliés au travers de 3 réseaux recouvrants. Chacun d'entre eux est utilisé pour la dissémination d'un type particulier de connaissances (haute, moyenne et basse). Le premier réseau est un réseau à 2 niveaux hiérarchiques. Le second est un réseau maillé avec un nombre de

réplicats k égal à 2 alors que le troisième est un réseau maillé avec un nombre de réplicats k égal à 1.

3.5.3 Évaluation de la dissémination des connaissances

Dans cette section, nous étudions la politique de dissémination de chacun des réseaux recouvrants considérés et nous les comparons sur la base de deux paramètres :

— Temps de propagation moyen d'une connaissance dans l'ensemble du réseau,
— Surcharge moyenne générée par la propagation d'une connaissance.

Chaque période de 5 secondes, un noeud du réseau génère un paquet de connaissances qu'il propage à l'ensemble du réseau. L'expérience est répétée de manière à ce que tous les noeuds du réseau puissent générer, un par un, de la connaissance.

La figure 3.19 représente le temps de propagation maximale d'une connaissance à l'ensemble des super noeuds du réseau pour chacun des 3 réseaux recouvrants étudiés. Le temps de propagation le plus court est obtenu par le réseau maillé avec k égal à 2 (0,74s). Ensuite, vient celui obtenu par le réseau hiérarchique à 2 niveaux (1,3s) et en dernier, celui généré par le réseau maillé avec k égal à 1 (2,1s). Les résultats obtenus par le réseau hiérarchique à 2 niveaux sont fortement corrélés à la topologie du réseau et à la position de l'hyper noeud.

La figure 3.20 représente la surcharge moyenne engendrée sur chaque lien du réseau par la dissémination d'une connaissance. Le réseau recouvrant qui génère le moins de surcharge sur chaque lien est le réseau hiérarchique à 2 niveaux (208 kbits). Le réseau maillé avec k égal à 1 obtient de moins bonnes performances (338 kbits), semblable à ceux obtenus par le réseau maillé avec k égal à 2 (695 kbits). Il s'avère en effet que les réseaux maillés génèrent beaucoup plus de surcharge. Le chemin emprunté par la connaissance n'est pas le plus court comme dans le cas de toute politique de type hot potato [Kaklamanis et al., 1993]. De plus, il arrive que la même connaissance soit envoyée à un noeud à plusieurs reprises pour les réseaux maillés avec $k = 2$.

Sur la base de ces résultats, le réseau utilisé pour la dissémination des connaissances de nature hautement prioritaire est constitué par le réseau maillé avec un nombre de réplicats k égal à 2. Pour les connaissances de

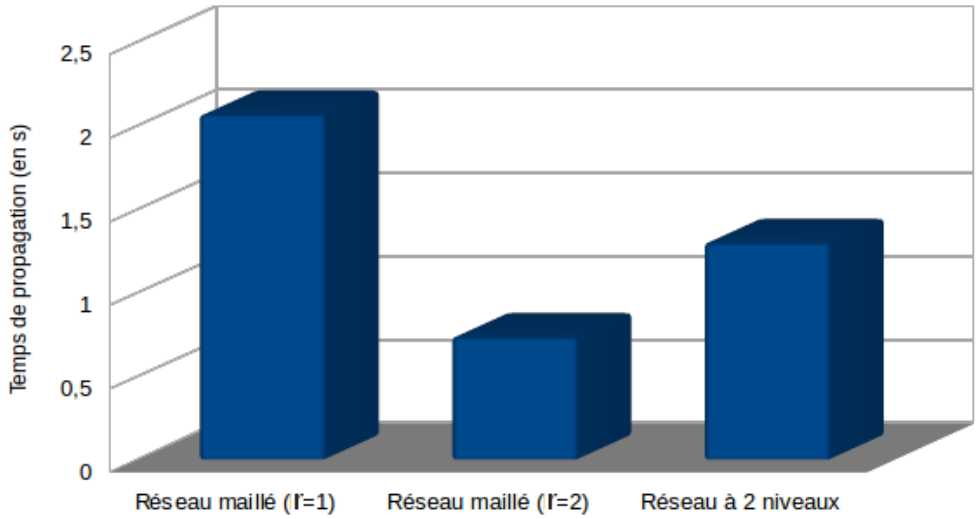


FIGURE 3.19 – Temps de propagation d'une connaissance à l'ensemble des noeuds du réseau.

moyenne priorité, il est conseillé de considérer un réseau à 2 niveaux hiérarchiques tandis que le réseau maillé avec un nombre de réplicats k égal à 1 est le plus approprié pour les connaissances de basse priorité. Nous utiliserons cette taxonomie dans la suite de notre scénario de simulation.

3.5.4 Cas applicatif : réaction à une attaque de type DDoS

Pour montrer un usage possible de notre plateforme de gestion de connaissances, nous évaluons deux possibles mécanismes de réaction à une attaque réseau de type déni de service distribué (DDoS), une fois l'attaque détectée. Le premier mécanisme consiste à bloquer le trafic incriminé au niveau de la cible tandis que le second consiste à bloquer le trafic incriminé au niveau du noeud le plus proche de la source. Dans ce travail, nous ne nous intéressons pas aux mécanismes de détection de l'attaque, mais uniquement à étudier l'intérêt de la dissémination des connaissances dans un cas d'usage réel.

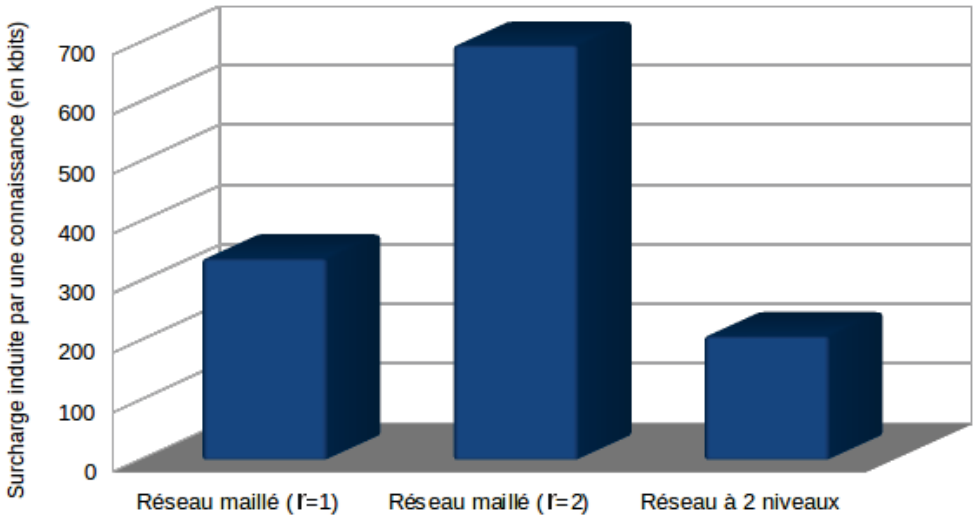


FIGURE 3.20 – Surcharge engendrée par une connaissance sur chaque lien du réseau.

3.5.4.1 Déni de service distribué (DDoS)

Une attaque par déni de service (DoS) représente une catégorie d'attaques visant à rendre un service, un serveur ou une infrastructure indisponible, en surchargeant intentionnellement la bande passante ou les ressources à disposition.

L'attaque par déni de service distribué est une forme plus évoluée d'une attaque DoS. Elle consiste à attaquer la cible à travers de nombreux points dans le réseau [Mirkovic and Reiher, 2004] (voir figure 3.21). Ces points, également appelés zombies, sont des machines qui ont été infectées et dont l'attaquant se sert pour attaquer sa cible. Plus le nombre de zombies est élevé, plus l'attaque de la cible est considérée comme violente.

Il existe plusieurs types d'attaques DDoS. Dans cette expérience, nous considérons une attaque par force brute de type inondation par paquets UDP. Elle consiste à envoyer des paquets UDP à la cible jusqu'à saturation de sa bande passante.

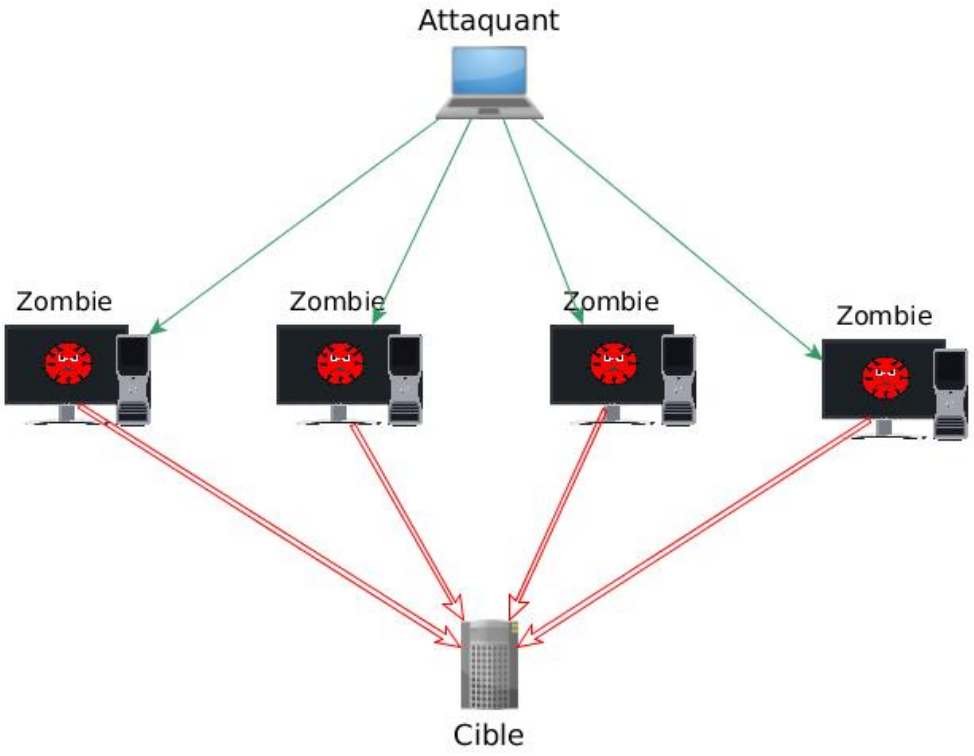


FIGURE 3.21 – Principe d'une attaque DDoS

3.5.4.2 Politiques de défense de la cible

Pour se défendre d'une attaque par force brute de type inondation par paquets UDP, la cible peut bloquer le trafic UDP incriminé via un pare-feu. Ce type de défense (action au niveau du serveur) protège la cible sans pour autant désengorger le réseau, qui reste quant à lui saturé.

Dans le cadre d'un réseau logiciel (SDN), nous proposons un mécanisme de protection en amont qui consiste à bloquer le trafic incriminé dès la source (voir figure 3.22).

Une fois l'attaque détectée au niveau du pare-feu de la cible, ce dernier bloque le trafic incriminé et propage ensuite sur l'ensemble du réseau les informations concernant les zombies participant à cette manoeuvre (toutes les machines du réseau possèdent une adresse IP publique). Ces informations atteignent les points d'accès auxquels sont connectés les zombies qui bloquent à leur tour le trafic incriminé (action au niveau des points d'accès). Il est

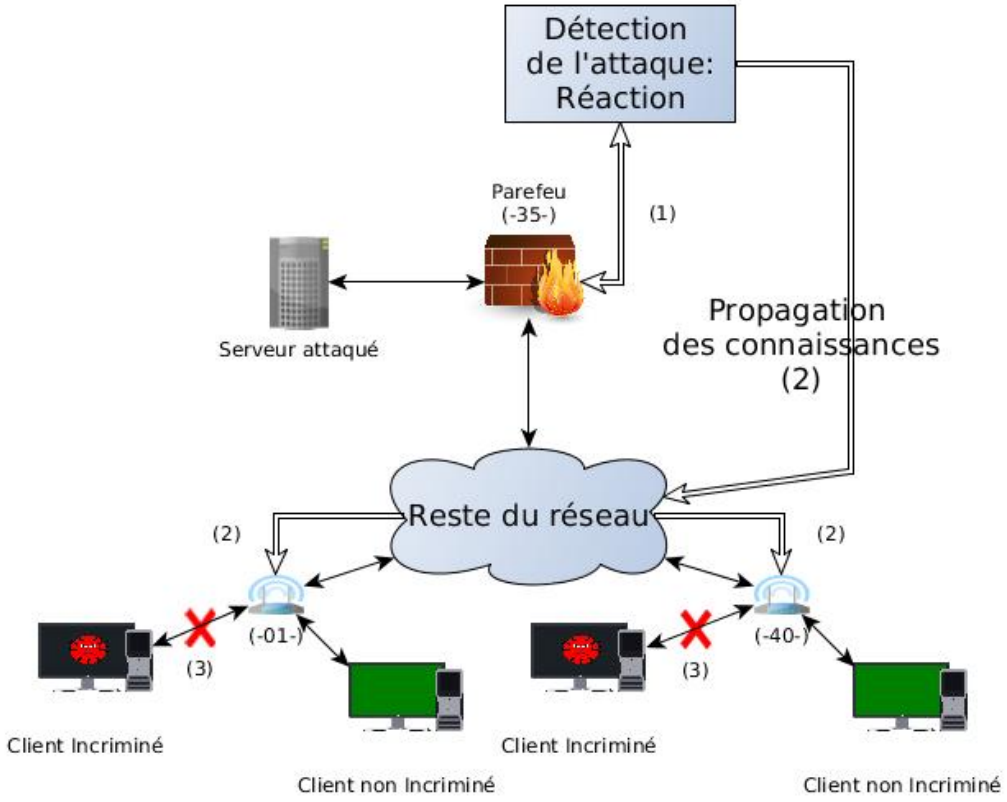


FIGURE 3.22 – Réaction du réseau à une attaque DDoS

évident que plus les informations sont transmises rapidement, plus les dommages impactant le réseau sont limités. Ce scénario permet de mettre en évidence l'importance d'un mécanisme de dissémination des connaissances utilisé pour transmettre ces informations, comme celui que nous avons proposé dans le cadre de cette thèse.

3.5.4.3 Évaluation de l'impact de la dissémination des informations

Dans cette section, nous définissons le scénario de simulation utilisé pour une attaque par force brute de type inondation de paquets UDP sur le réseau NTT décrit dans la section 3.5.1. La cible de l'attaque est une machine reliée au noeud 35 qui fait office de pare-feu [8]. Les points d'accès sélectionnés correspondent aux noeuds 1, 3, 5, 13, 31, 28, 32, 54, 53, 48, 44 et 37. Ils sont

8. Le pare-feu considéré est Netfilter (http ://netfilter.org/).

munis chacun d'un pare-feu. Une machine zombie est ensuite reliée à chacun de ces points. Chaque période de 10 secondes, un zombie supplémentaire est activé et génère un trafic UDP de 10 Mbits/s.

Le noeud 35 détecte l'attaque au bout du troisième zombie activé. Dans la première politique de défense de la cible (action au niveau du serveur), il se contente de bloquer le trafic UDP à son niveau. Dans la seconde politique (action au niveau des points d'accès), il dissémine les informations relatives aux zombies attaquants sur le réseau en utilisant notre plateforme de gestion de connaissances. Chaque point d'accès informé de la présence d'un zombie qui lui est rattaché bloque le trafic UDP à son niveau.

La figure 3.23 montre l'impact sur le réseau des deux politiques de défense de la cible. Elle représente le trafic malveillant circulant sur le réseau en fonction du temps. Ce trafic est évalué par périodes de 5 secondes.

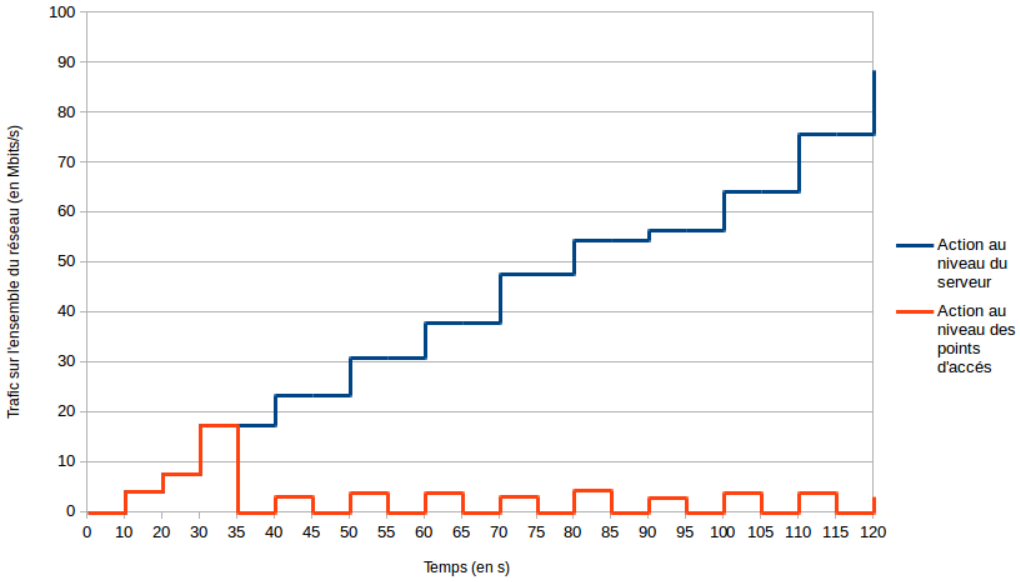


FIGURE 3.23 – État du réseau dans le cas d'une attaque DDoS

La figure 3.23 montre qu'à partir du moment où l'attaque est détectée, chaque zombie activé est bloqué par son point d'accès dès que celui-ci est informé, d'où la disparition immédiate du trafic malveillant que le zombie a généré. L'intérêt d'une approche basée sur l'usage d'un plan de connaissance dans ce type de scénario est important. Cette expérience a servi à démontrer que plus la dissémination est rapide, moins la surcharge engendrée sur le réseau

liée au trafic malveillant est importante.

3.6 Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps la conception de notre plateforme de gestion de connaissances, nommée Autolay. Celle-ci se base sur une topologie hiérarchique à base de super noeuds et d'hyper noeuds. Nous avons comparé notre algorithme d'élection des noeuds, DCLARA, avec un algorithme centralisé SPSP (Super-Peer Selection Problem). Les résultats obtenus montrent que l'approche proposée améliore la mesure de dissimilarité totale du système.

Nous avons ensuite évalué notre plateforme en simulation. Cette évaluation montre clairement que les approches que nous avons proposées (DCLARA et DCLARA-PH) offrent de meilleurs résultats comparativement aux approches à base de tables de hashage distribuées ou aux approches gloutonnes. Les approches hiérarchiques, sur lesquelles se basent les solutions proposées, sont tout à fait adaptées au problème de la diffusion des connaissances dans un grand réseau. Seulement, le coût de maintien de ce type de structure et de son efficacité au fil du temps peut être problématique dans un réseau dont la dynamicité est élevée. L'utilisation du test de page Hinkley, en plus d'améliorer les performances dans le contexte d'une topologie réseau fortement dynamique, permet de répondre à cette problématique.

Par ailleurs, nous avons évalué la plateforme proposée dans un environnement virtualisé. Dans l'objectif d'améliorer le temps de dissémination des connaissances dans le réseau, nous les avons classifié en 3 types (connaissance à priorité haute, à priorité moyenne et à priorité basse) et nous avons comparé les différents types de réseaux recouvrants implémentés afin de sélectionner le plus approprié à la dissémination de chaque type de connaissance.

Dans un but de montrer les usages possibles de notre plateforme dans le cadre des réseaux logiciels (Software Defined Network), nous avons mis en oeuvre deux cas d'usage complètement différents. Le premier concerne la reconfiguration dynamique des réseaux recouvrants de diffusion de contenus. Le second traite d'un problème de sécurité dans les réseaux qui a pour objectif de montrer l'apport de la dissémination des connaissances dans une politique de réaction à une attaque de type déni de service distribuée.

Afin de mettre en évidence l'aspect générique de notre plateforme, nous

étudions, dans le chapitre suivant, deux exemples d'applications liés au Cloud Computing et aux MicroGrids.

Chapitre 4

Cas d'usage de la plateforme de gestion de connaissance : Cloud Computing et MicroGrids

Sommaire

4.1 Introduction

Pour montrer l'intérêt de notre plateforme de gestion de connaissance, nous proposons deux cas d'usage. Dans le premier, nous avons proposé un mécanisme adaptatif et en temps réel de sélection des fournisseurs IaaS afin de satisfaire au mieux les besoins des utilisateurs. Les résultats montrent clairement une amélioration des performances par rapport à une approche non adaptative. Dans le second cas, nous démontrons que notre plan de connaissance peut être utilisé dans le contexte des SmartGrids (réseaux intelligents de distribution d'énergies).

4.2 Sélection adaptative d'un fournisseur Cloud IaaS

Traditionnellement, les entreprises hébergent en interne leurs propres services, plateformes et infrastructures sur leurs propres serveurs ou en partageant un datacenter avec d'autres entreprises. Cette politique engendre des coûts élevés à la fois en prix d'achat du matériel et en ressources humaines (maintenance et entretien). Elle peut être aussi inadaptée (souvent surdimensionnée) aux besoins réels de l'entreprise. Dans ce contexte, l'une des solutions est d'utiliser le "Cloud Computing" pour externaliser leurs services. Celui-ci se définit par la mise à la disposition du client Cloud des machines très performantes en termes de puissance de calcul et de capacité de stockage, et d'une bande passante importante. Le Cloud se définit aussi par la location d'infrastructures logicielles et matérielles aux clients selon leurs besoins.

Le Cloud Computing est rendu possible grâce à l'amélioration des infrastructures des réseaux informatiques. En effet, les connexions à haut débit ont permis de réduire la latence à un niveau acceptable et donc de rendre possible l'utilisation de ressources distantes. Le succès du Cloud Computing auprès des entreprises a conduit non seulement à la réussite et à l'expansion de ce concept, mais aussi à l'explosion du nombre de fournisseurs de Cloud computing proposant des offres multiples et variées. L'accès à ces services se fait par le biais d'un abonnement. Seulement, il n'existe aujourd'hui aucune technique permettant de s'abonner, au plus juste, à des opérateurs Cloud, et par extension, de sélectionner le fournisseur Cloud le mieux adapté à ses besoins.

La sélection d'un fournisseur est un problème d'optimisation souvent étudié dans plusieurs domaines. Étant donné le nombre important de paramètres et d'acteurs à prendre en compte dans le contexte du Cloud, ce problème est réputé NP-complet. Cette section décrit une application directe de nos travaux dans ce domaine. Nous proposons, ici, une nouvelle méthode de sélection de fournisseurs Cloud, adaptative et en temps réel, qui s'appuie sur l'utilisation de notre plateforme.

Avant de développer la méthode proposée, nous décrirons dans la suite le concept du Cloud et positionnons notre approche dans ce contexte.

4.2.1 Le Cloud Computing

Le Cloud computing ou "informatique dans les nuages" est le terme utilisé pour désigner la mise à disposition de ressources informatiques distantes par le biais du réseau. Ces ressources sont fournies sous forme de services à la demande accessibles depuis des ordinateurs fixes ou portables, des smartphones, des télévisions connectées ou des tablettes [Bhardwaj et al., 2010]. Ces ressources peuvent être logicielles comme des plateformes de développement ou de travail collaboratif ou bien matérielles comme des serveurs de stockage, des serveurs de calcul ou des équipements réseau (routeur, pare-feu, ...).

Selon les recommandations du NIST (National Institute of Standards and Technologies) [Mell and Grance, 2011], le Cloud Computing doit être caractérisé par :

— La mise à disposition des services à la demande ou au self-service : les utilisateurs ont un accès immédiat et automatique à des puissances de calcul et des capacités de stockage sans passer par l'intermédiaire d'une personne physique [Calheiros et al., 2011].
— L'accessibilité via Internet : toutes les ressources offertes par le Cloud Computing sont accessibles via le réseau Internet à travers des mécanismes de connexion standards HTTP et à partir d'appareils hétérogènes, ordinateurs fixes ou mobiles, smartphones, télévisions connectées et tablettes. L'utilisateur n'aura qu'à se connecter via un navigateur web ou une application dédiée pour avoir accès à son compte Cloud.
— Le partage de ressources : les ressources des fournisseurs du Cloud Computing sont partagées entre les utilisateurs selon les besoins de chacun d'entre eux.
— L'élasticité : l'allocation des capacités de stockage et des puissances de calcul peuvent être modifiées par l'utilisateur à la volée pour s'adapter à l'évolutivité des besoins du client.
— Des services mesurables : les systèmes du Cloud Computing doivent contrôler et optimiser automatiquement les ressources à travers des métriques telles que la capacité de stockage, la puissance de calcul, la bande passante ou le nombre des utilisateurs actifs.

4.2.2 Modèles des services du Cloud Computing

Les services fournis par le Cloud Computing peuvent être catégorisés en trois types [Mell and Grance, 2011] : SaaS (Software as a service), PaaS (Plat-

form as a service) et Iaas (Infrastructure as a service).

4.2.2.1 SaaS

Le logiciel en tant que service est défini par la capacité offerte aux utilisateurs d'exploiter des logiciels installés sur les infrastructures du fournisseur Cloud. Généralement, ces logiciels sont accessibles via un simple navigateur Internet [Iosup et al., 2011]. Les utilisateurs n'ont pas à se soucier de l'infrastructure Cloud sous-jacente. Par conséquent, ils ne gèrent ni la maintenance du logiciel ni le serveur qui l'héberge.

Le service de messagerie Gmail est, par exemple, un service SaaS fourni par Google gratuitement à ses clients. L'utilisateur se connecte à son compte et accède instantanément à toutes les fonctionnalités disponibles sur ce service. Toute l'exécution de l'application se fait du côté du serveur et la partie client ne prend en charge que l'affichage et la présentation [Keller and Rexford, 2010].

Le concept du SaaS a changé la façon dont les applications sont développées et fournies aux utilisateurs. Cependant, ce modèle de service présente plusieurs risques au niveau de la sécurité et de la confidentialité des données. En effet, l'utilisateur du SaaS n'a aucune idée sur le fonctionnement interne du logiciel qu'il utilise.

4.2.2.2 PaaS

La plateforme en tant que service est définie par la capacité offerte aux utilisateurs de déployer leurs propres applications sur les infrastructures du fournisseur Cloud. Les utilisateurs ne gèrent que la maintenance de leurs applications sans se préoccuper du serveur qui les héberge.

Le service PaaS facilite aux développeurs le développement et le déploiement d'applications sans avoir à se soucier du coût et de la complexité liés à la gestion de la plateforme de développement ou d'hébergement. L'un des avantages d'utilisation du service PaaS est la simplicité de la phase du développement. En effet, ce service permet aux développeurs d'utiliser une seule plateforme fournissant tous les utilitaires requis de la création à la délivrance d'une application (outils de développement, outils de test et outils de déploiement) et réduisant ainsi le temps de développement et de déploiement.

Google AppEngine est un exemple d'un service PaaS qui permet au développeur de coder une application web en langage Java ou en langage Python. EngineYard est également un service PaaS qui permet de développer une application en langage Ruby.

Cependant, les temps d'apprentissage et d'adaptation nécessaires pour la prise en main de la plateforme peuvent constituer un inconvénient dans l'utilisation des services de ce type [Huppler, 2012].

4.2.2.3 IaaS

L'infrastructure comme service consiste à fournir des ressources matérielles (serveurs, équipements réseau) et des ressources logicielles (systèmes d'exploitation, bases de données) sous la forme de services. Contrairement aux formules d'hébergement traditionnelles où l'allocation des ressources est rigide, le service IaaS permet aux utilisateurs d'acquérir à la volée des ressources selon leurs besoins évolutifs. En effet, le service IaaS est fourni avec un contrat qui indique que le client ne paie que ce qu'il a consommé et exclut tout engagement ferme [Gillam et al., 2013].

Les fournisseurs de IaaS ne gèrent pas les systèmes d'exploitation et les applications déployées par l'utilisateur, mais uniquement le bon fonctionnement de leur datacenter.

Amazon Web Service (AWS), Elastic compute Cloud (EC2) et Secure Storage Service(S3) sont des exemples de fournisseurs IaaS [Gonçalves and Ballon, 2011].

4.2.3 Problématique du choix d'un fournisseur Cloud

Devant la difficulté de comparaison des fournisseurs de services SaaS ou PaaS (chaque logiciel ou plateforme ayant des caractéristiques spécifiques), nous nous sommes contentés, dans ce travail, d'étudier l'accès aux services IaaS fournis par les opérateurs Cloud.

Le nombre de fournisseurs IaaS est tel qu'il est difficile d'être certain de faire le bon choix. Si pour une utilisation personnelle, le choix du fournisseur a un impact mineur, il en est tout autre pour les entreprises. En effet, face à des marchés fortement compétitifs, les entreprises réalisent aujourd'hui qu'une gestion efficace de leurs achats peut constituer un avantage concurrentiel sub-

stantiel [Aguezzoul et al., 2006]. La sélection des fournisseurs devient ainsi une décision stratégique qui a un impact crucial sur la performance globale de toute l'entreprise.

La sélection d'un fournisseur de type Cloud Computing est donc une tâche délicate. De plus, la diversité des critères de mesure de performance rend cette tâche encore plus difficile.

4.2.3.1 Évaluation des fournssieurs IaaS

La qualité de la sélection des fournisseurs de Cloud Computing dépend fortement à la fois des critères de performance utilisés et de la qualité des mesures collectées. Des informations insuffisantes, erronées, imprécises ou non actualisées conduisent, en effet, à une mauvaise prise de décision. Partant de ce constat, nous nous sommes basés sur nos travaux concernant le plan de connaissance que nous avons mis en place afin d'assurer les tâches de collecte et de dissémination de ces informations. Nous avons conçu des agents capables de calculer plusieurs indicateurs objectifs [Li et al., 2010] mesurant la qualité des fournisseurs du marché (voir Table 4.1). Ces informations, relatives à la fois au réseau, au stockage, à la capacité de calcul et au coût financier, sont récupérées d'une manière régulière et alimentent en continu notre base de connaissances.

Catégories	Métriques
Réseau	Latence, débit, disponibilité, bande passante.
Stockage	Temps de réponse, débit.
Capacité de calcul	Performance CPU.
Prix	Tarif appliqué par le fournisseur

TABLE 4.1 – Mesures de performance

4.2.3.2 Fournisseurs IaaS étudiés

Afin de valider notre approche, nous avons considéré 4 fournisseurs parmi les plus connus et les plus influents sur le marché : Amazon, HP, Microsoft Azure et Numergy. Nous avons créé une instance IaaS (une machine virtuelle) chez chacun de ces fournisseurs. Ces instances sont petites et de capacité théorique égale [1]. Les capacités de chaque instance ainsi que leur coût mensuel d'utilisation à temps plein sont représentés par le tableau 4.2.

1. Selon les données fournies par les fournisseurs.

Fournisseur	Amazon	HP	Microsoft	Numergy
Nombre de coeur de calcul virtuel (vCPU)	1 coeur	1 coeur	1 coeur	1 coeur
Espace de stockage disponible	8GB	8GB	30GB	8GB
Quantité de mémoire vive (RAM)	1 GB	1 GB	1 GB	1 GB
Coût mensuel d'une utilisation à temps plein	43.20€	49.90€	43.80€	35.77€

TABLE 4.2 – Caractéristiques des 4 instances

4.2.3.3 Évaluation des paramètres réseau

Étant donné que les infrastructures sont situées dans des datacenters distants, la qualité de service offerte par un fournisseur donné sera dépendante de la qualité du réseau qui relie le client au datacenter. Cette dépendance induit une forte corrélation aux paramètres réseau dans l'évaluation d'un fournisseur Cloud [Kuo et al., 2010]. Notre plan de connaissance doit donc contenir assez d'information ces paramètres pour pouvoir se prononcer sur le meilleur fournisseur. Afin d'évaluer ces derniers, nous avons mis en place des agents qui évaluent la bande passante et la latence d'une connexion en téléchargeant des données multimédias volumineuses (voir figure 4.1).

De plus, pour évaluer l'impact de la position géographique d'un utilisateur sur les performances du réseau, notre plan de connaissance doit être alimenté par des mesures effectuées à partir de plusieurs sources reparties à travers le réseau Internet. Pour ce faire, des agents ont été répartis sur un ensemble de noeuds du réseau PlanetLab [Culler et al., 2002]. PlanetLab est un réseau mondial de recherche qui soutient le développement de nouveaux services de réseau créé en 2003. Il est actuellement composé de 1170 noeuds répartis sur 561 sites.

L'évaluation des performances réseau est assurée par une tâche cyclique exécutée toutes les 30 secondes par chaque agent. La durée de la compagne de mesure est d'une journée entière (24h). Ensuite, chaque agent analyse et agrège les informations collectées avant de les stocker dans la base de connaissance (KB). L'ensemble des KBs est ensuite synchronisé en utilisant la plateforme que nous avons mise en place. Ces informations forment le plan de connaissance.

En raison du nombre important des pays concernés par la campagne d'éva-

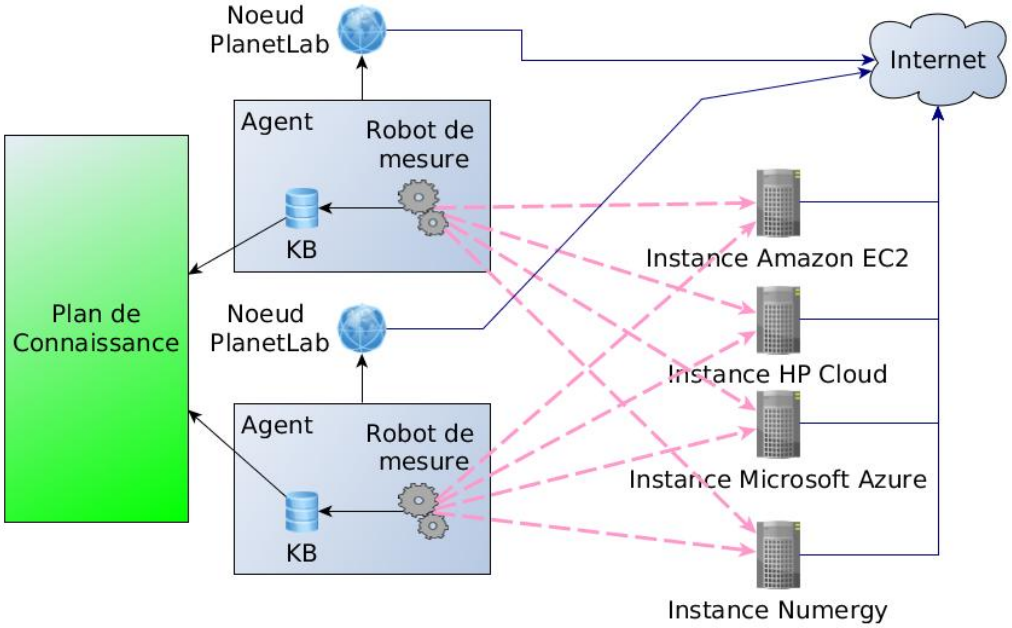
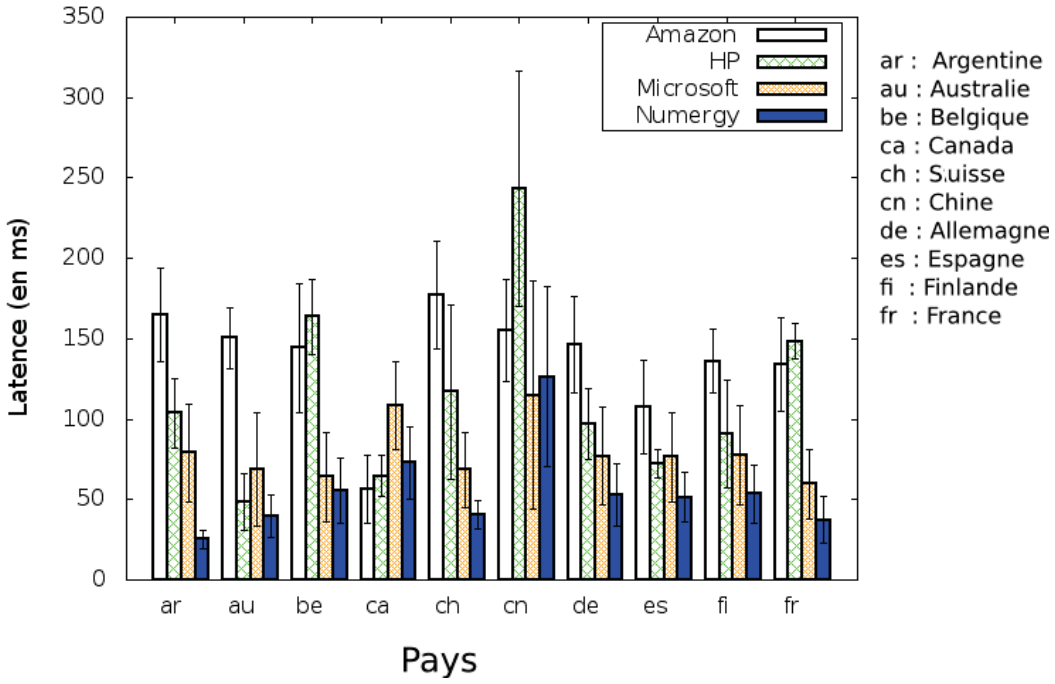


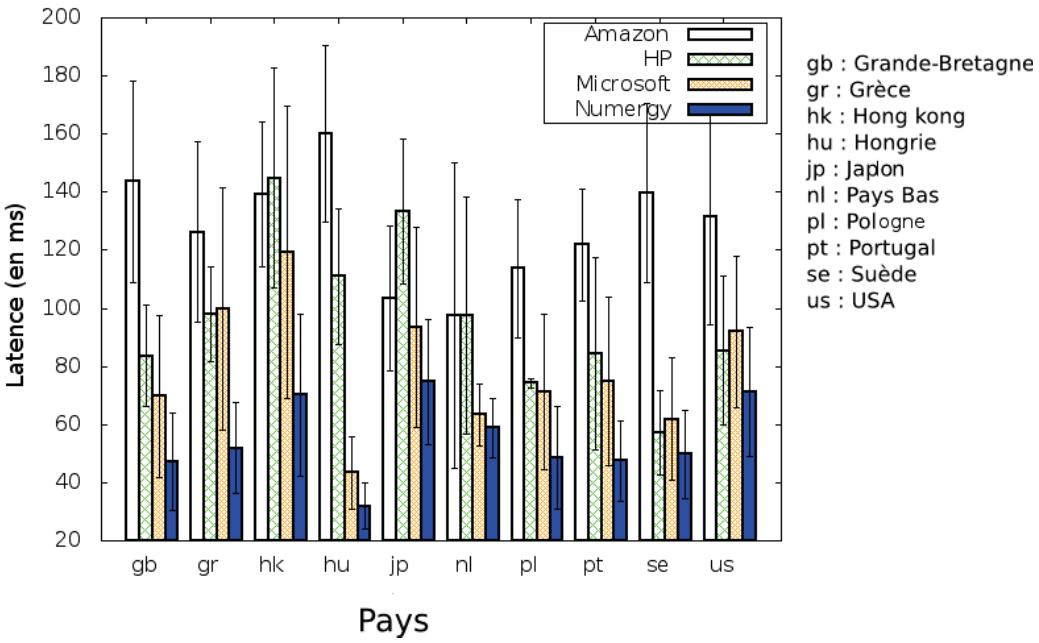
FIGURE 4.1 – Collecte des informations réseau

luation, les résultats seront exposés sur deux figures pour plus de clarté.

La figure 4.2 montre les variations de la latence (en millisecondes) sur une journée de mesure pour les quatre fournisseurs de Cloud par pays tandis que la figure 4.2.3.3 synthétise les variations de la bande passante (en kbps).

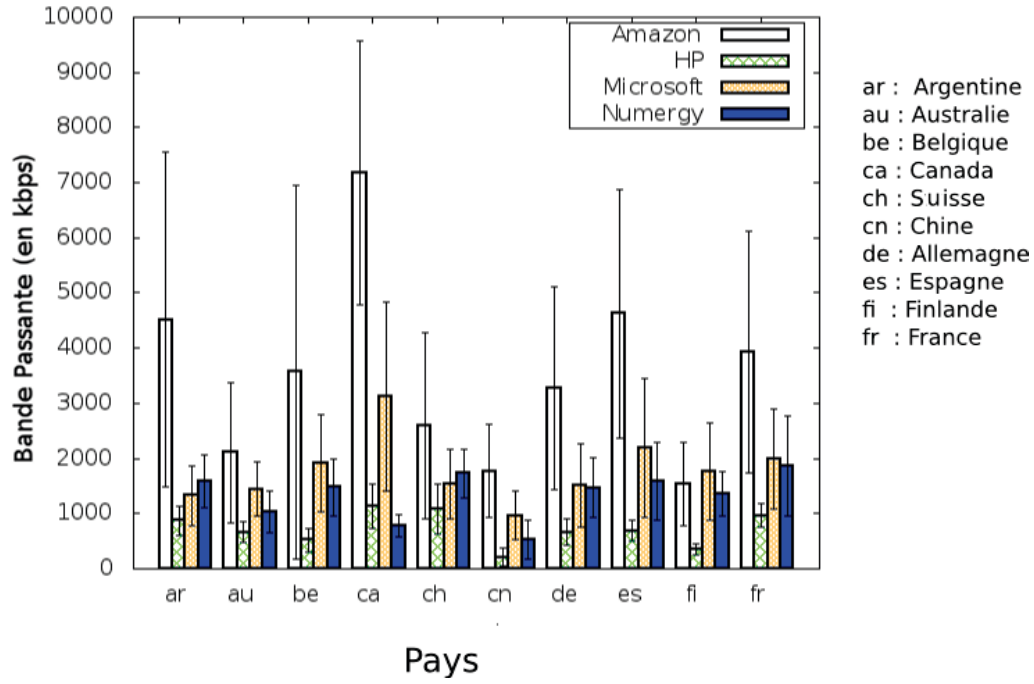


(a)

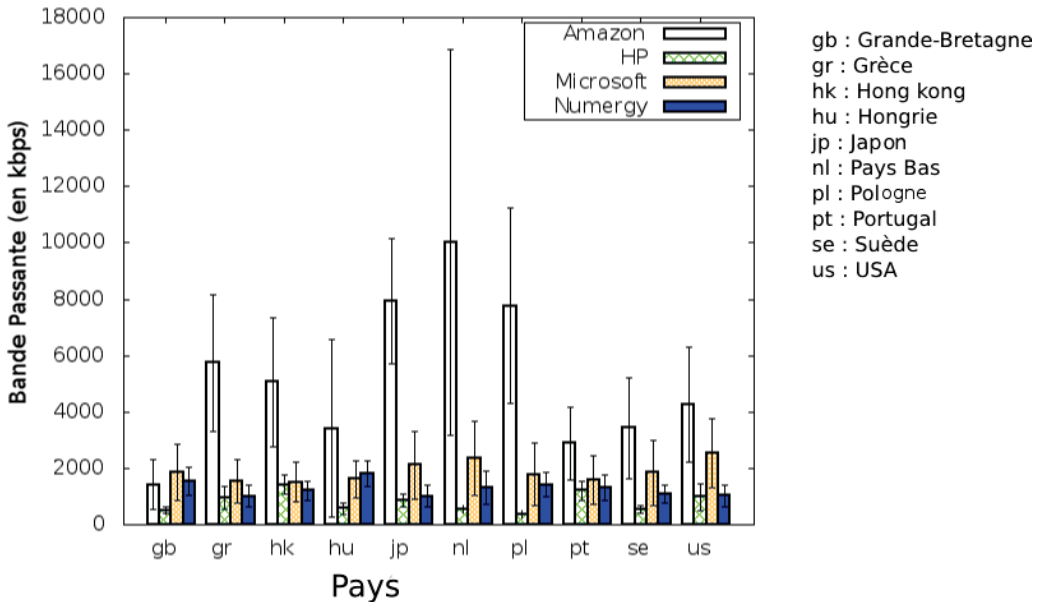


(b)

FIGURE 4.2 – Latence moyenne et écart type par pays



(a)



(b)

FIGURE 4.3 – Bande passante moyenne et écart type par pays

4.2.3.4 Évaluation des paramètres de calcul et de stockage

Pour évaluer les performances matérielles, nous avons déployé des robots sur les 4 instances créées. Ces robots seront interrogés par un agent à travers une interface web afin d'alimenter notre plan de connaissance (cf. figure 4.4). L'objectif est de fournir un indicateur de base sur la performance d'un système de type Linux. De multiples tests sont effectués afin de tester divers aspects de la performance du système.

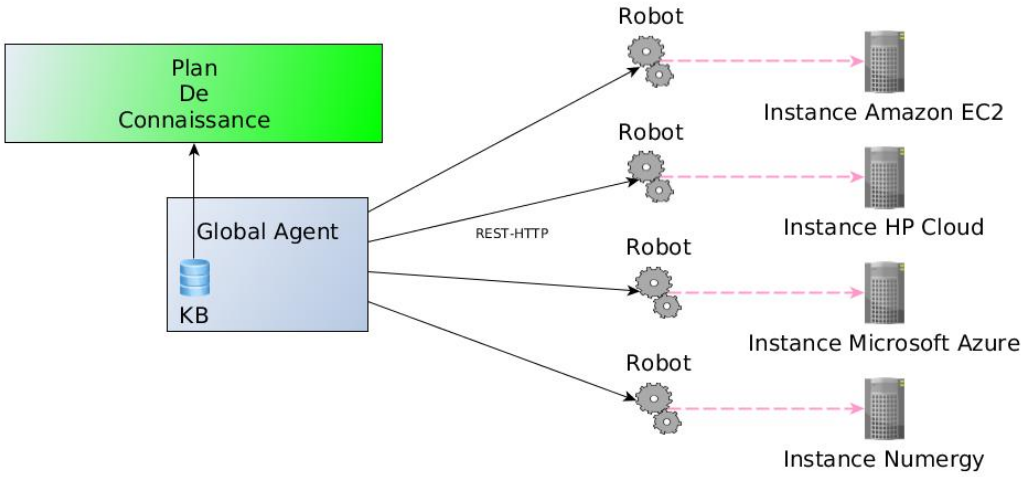


FIGURE 4.4 – Collecte des informations système

Les robots sont basés sur l'outil d'UnixBench [Hatt et al., 2007] qui représente une référence dans la mesure des systèmes Linux et Unix. Le résultat obtenu par UnixBnech ne dépend pas seulement des ressources matérielles (CPU, RAM et disque) mais aussi de la nature du système d'exploitation, des librairies et du compilateur installé sur la machine. Pour obtenir des résultats objectifs, nous avons donc pris soin d'installer les mêmes composants logiciels sur les 4 instances considérées.

UnixBench est composé de différents tests décrits ci-dessous :

Dhrystone : Ce benchmark est utilisé pour mesurer et comparer les performances des ordinateurs en effectuant des tests sur la gestion des chaînes de caractères. Ce test constitue un indicateur de performance à la fois sur la puissance de calcul, sur la vitesse de transfert de la mémoire vive et de la mémoire cache.

Whetstone : Ce test mesure la vitesse et l'efficacité des opérations en virgule

Fournisseur	Amazon	HP	Microsoft	Numergy
Score de la mesure comparative	71,5	1398,7	762,0	1724,4

TABLE 4.3 – Score UnixBench agrégé pour chaque fournisseur

flottante. Ce test comporte plusieurs modules censés représenter une combinaison d'opérations habituellement réalisées dans les applications scientifiques. Il s'agit d'un indicateur sur la puissance de calcul.

Execl Throughput : Ce test mesure le nombre d'appels "Execl" qui peuvent être effectués par seconde. "Execl" fait partie de la famille de fonctions "exec" qui remplace l'image du processus appelant par une nouvelle image du processus. Il s'agit d'un indicateur sur la puissance de calcul et de la mémoire vive.

File copy : Ce test permet de mesurer la vitesse à laquelle les données peuvent être transférées d'un fichier à un autre, en utilisant différentes tailles de mémoire tampon. Ceci donne une idée sur la performance du disque dur.

Pipe Throughput : Un tube (*pipe*) est la forme la plus simple de communication entre les processus. Le débit d'un tube détermine le nombre de fois qu'un processus puisse écrire ou lire 512 octets sur un tube chaque seconde. Il s'agit d'un indicateur sur la vitesse d'accès à la mémoire vive.

Pipe-based Context Switching : Ce test mesure le nombre de fois où deux processus peuvent échanger un nombre entier à travers un tube chaque seconde. Il s'agit d'un indicateur sur la vitesse d'accès à la mémoire vive.

Process Creation : Ce test mesure le nombre de fois où un processus peut enfanter. Il s'agit d'un indicateur sur la puissance de calcul et de la mémoire vive.

Shell Scripts : Mesure le temps d'exécution d'un script en simulant des opérations courantes. Il s'agit d'un indicateur sur la puissance de calcul.

System Call Overhead : Mesure le temps de latence induit par un appel système. Il s'agit d'un indicateur sur la puissance de calcul.

Les résultats de ces tests sont ensuite comparés aux scores d'un système de référence pour produire une valeur comparative. L'ensemble de ces valeurs est ensuite combiné afin de produire un score global pour chaque instance (voir le tableau 4.3).

Théoriquement, les 4 instances doivent disposer de performances quasiment identiques (voir le tableau 4.2). Or, en réalité, il y a une forte disparité

entre elles. Celle-ci est due, principalement, aux ressources matérielles utilisées par chacun des fournisseurs Cloud.

4.2.4 Proposition d'un mécanisme de sélection pour le Cloud

Après cette phase d'évaluation des fournisseurs Cloud, notre plan de connaissance sera en charge de l'analyse et du traitement des informations collectées. L'objectif est de pouvoir répondre à la demande d'un utilisateur qui souhaite obtenir le fournisseur le plus adéquat à ses besoins. En effet, un fournisseur X peut être le meilleur choix pour le client 1 et ne pas l'être pour un autre client. De plus, ce choix peut évoluer dans le temps.

Notre première approche, nommée SCPS (**S**tatic **C**loud **P**rovider **S**election), est basée sur les techniques de programmation linéaire qui consistent à trouver une solution optimale en maximisant une fonction de coût donnée (cf. équation 4.1).

$$sel(U, P) \rightarrow Max_{(i=0 \rightarrow n)}(F_i(x_1, x_2, ..., x_k)) \tag{4.1}$$

Dans notre cas, la fonction de coût prend en considération 4 paramètres :

— La bande passante
— La latence
— Le prix
— La performance des instances.

Toutefois, et au regard du nombre des paramètres pris en considération, la détermination de la fonction de coût est difficile. De plus, cette fonction est dépendante des besoins réels de chaque utilisateur qui évoluent au cours du temps. Nous faisons ici le choix de laisser à l'utilisateur la liberté de définir sa propre fonction de coût pour qu'elle soit une combinaison linéaire de ses 4 paramètres comme le montre la figure 4.5. Par exemple, si l'utilisateur souhaite déployer un service dans lequel le transfert de fichiers est important, il va donner une grande importance à la bande passante. S'il souhaite déployer un service nécessitant un besoin de communication en temps réel, il donnera plus d'importance à la latence.

Dans le cas où l'utilisateur ne définit pas sa propre fonction de coût, l'équation considérée sera celle décrite par 4.2. La bande passante, la latence, le prix

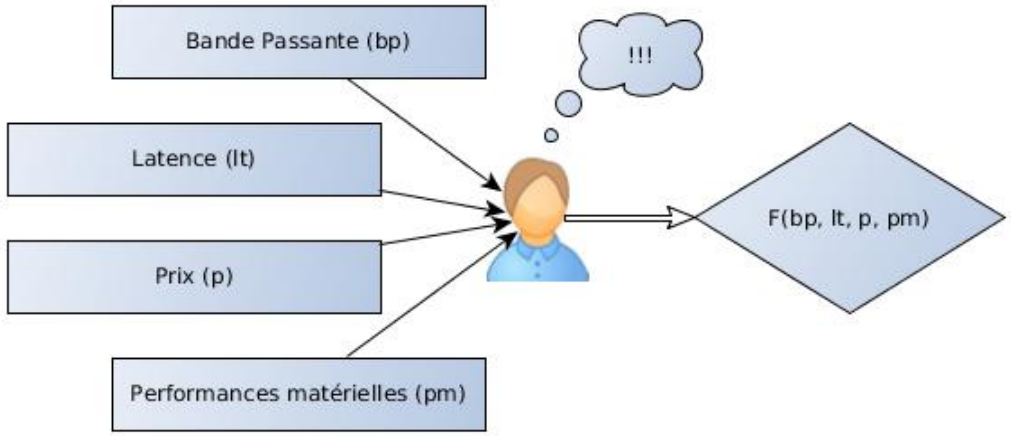


FIGURE 4.5 – Détermination de la fonction de coût

et la performance du système sont normalisés en utilisant la norme infinie. A chacun de ces paramètres, l'utilisateur doit affecter un poids x_i selon ses préférences. Ces paramètres sont ensuite pondérés pour obtenir un score final agrégé.

$$\begin{aligned} bp &= \frac{\bar{bp}+std(bp)}{\|bp\|_\infty} \\ lt &= \frac{\|lt\|_\infty}{\bar{lt}+std(lt)} \\ p &= \frac{\|p\|_\infty}{\bar{p}+std(p)} \\ pm &= \frac{\bar{pm}+std(pm)}{\|pm\|_\infty} \\ F(bp,lt,p,pm) &= x_1.bp + x_2.lt + x_3.p + x_4.pm \end{aligned} \tag{4.2}$$

Afin de prendre en considération la localisation géographique des usagers finaux du service à déployer, nous prenons en considération les performances réelles mesurées pour cette localisation. Cependant, si la localisation n'est pas disponible dans les données dont nous disposons, nous prenons en considération les mesures de la localisation la plus proche (voir l'équation 4.3).

$$loc(U,P) = Min_{(i=0\rightarrow n)}d(ULoc - PLoc_i) \tag{4.3}$$

Si plusieurs localisations sont à considérer, les mesures sont pondérées selon les taux de répartition des usagers afin d'obtenir des mesures globales.

Afin de prendre en considération les variations des performances des fournisseurs, nous avons proposé une seconde approche nommée ACPS (**A**daptive

Cloud **P**rovider **S**election). Cette dernière consiste à alimenter en continu le plan de connaissance afin de pouvoir choisir le fournisseur le plus adapté aux besoins de l'utilisateur à chaque instant. Le changement de fournisseur s'effectue sur la base de l'algorithme de Page-Hinkley [Sebastião and Gama, 2009] (cf. section 2.7.2) comme le montre l'équation 4.4 :

$$\begin{aligned} &\bar{d}_T = \sum_{t=1}^{T} d_t \ / \ \mathrm{T} \\ &X_T = \sum_{t=1}^{T} (d_t - \bar{d}_T) \\ &\text{if } X_T > \lambda \rightarrow \text{re-choisir le fournisseur le plus adapté.} \end{aligned} \tag{4.4}$$

d_t représente le résultat de la fonction de coût à un instant t, $\bar{d}_T$ la moyenne des d_t et X_T la différence cumulée entre d_t et $\bar{d}_T$. Lorsque X_T est supérieur à un seuil λ donné, un point de changement est détecté.

4.2.4.1 Étude du cas d'un client

Pour évaluer les performances de nos deux approches (l'approche adaptative ACPS et l'approche statique SCPS), chaque agent va jouer le rôle d'un *broker*[2]. Sur la base du plan de connaissance construit et des besoins d'un client, il sélectionne dynamiquement le fournisseur Cloud le plus adéquat.

Le cas étudié est un client qui possède 50 % de son activité aux États-Unis, 30 % en France et 20 % dans d'autres pays et qui considère que les 4 paramètres (la bande passante, la latence, le prix et la performance du système) ont la même importance.

La figure 4.6 montre la variation du score obtenu par les deux approches SCPS et ACPS :

— La méthode SCPS, dite méthode statique, consiste à sélectionner le meilleur fournisseur à un instant t (fixé à 4 heures dans cette expérimentation) et de le garder toute la journée.
— La méthode ACPS consiste à décider d'un abonnement à la volée et à changer de fournisseur à chaque fois que ceci est nécessaire. Le paramètre λ est fixé à $0,6$.

La forte variation des scores de l'approche SPCP nous laisse penser qu'il est possible d'améliorer la fonction de coût en faisant une sélection adapta-

2. Selon le NIST [Mell and Grance, 2011], un *broker* est une entité qui fait office d'intermédiaire entre les fournisseurs et les consommateurs du Cloud en gérant l'utilisation et l'évaluation des services fournis.

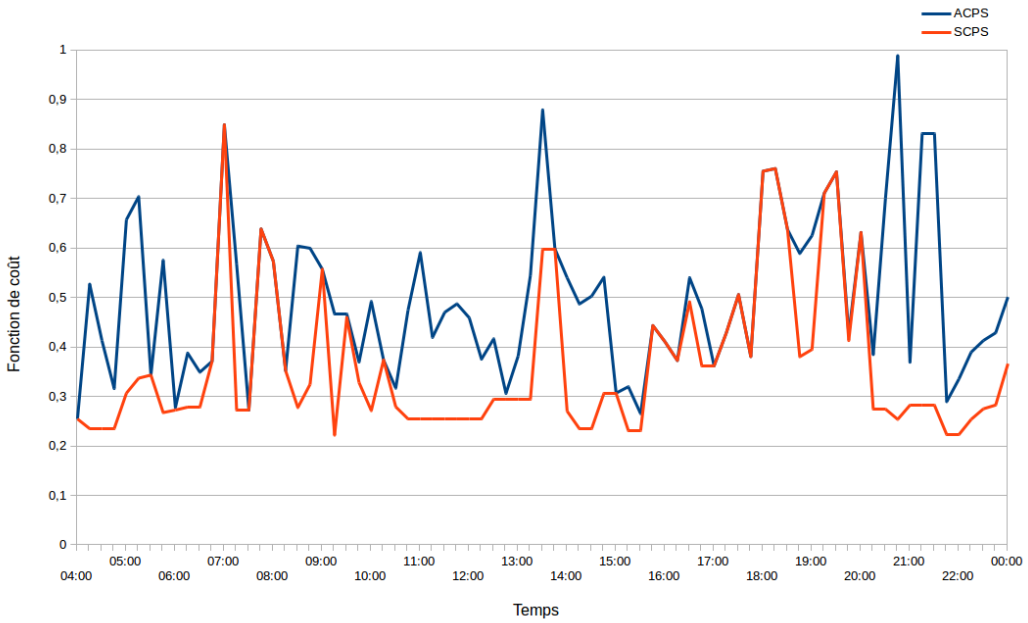


FIGURE 4.6 – Sélection d'un fournisseur

tive des fournisseurs et en passant de l'un à l'autre selon celui qui obtient le meilleur résultat. L'approche adaptative que nous avons mise en place donne un meilleur choix dans le temps et représente à chaque fois le meilleur coût. Cette nouvelle approche de sélection de fournisseurs de Cloud permet, sur la base du plan de connaissance construit et partagé à travers notre plateforme, de nous donner la possibilité de vérifier les capacités des fournisseurs en temps réel et de migrer chez un autre fournisseur à chaque fois que l'offre proposée voit ses performances se dégrader ne répondant plus ainsi aux spécifications originelles du client.

4.3 Gestion du flux de connaissances au sein des MicroGrids

Les MicroGrids, au même titre que les PowerGrids, sont des réseaux de production et de distribution d'énergie. Contrairement à ces derniers, ils sont de taille plus modeste et sont surtout caractérisés par des facultés d'auto-gestion des phases de production, de distribution et de consommation d'énergie. L'objectif principal consiste à satisfaire au mieux les exigences des usagers

en matière d'énergie sans qu'aucune intervention humaine ne soit requise. Les propositions de MicroGrids existantes utilisent un agrégateur central ou, dans le meilleur des cas, partent du principe qu'il existe une plateforme distribuée pour gérer les informations nécessaires à son bon fonctionnement. Nous proposons de déployer notre plateforme comme un moyen de collecter, gérer et distribuer la connaissance dans les MicroGrids.

4.3.1 Des PowerGrids vers les MicroGrids

Le rôle principal d'un PowerGrid est de connecter les producteurs d'énergie ou MacroGrids (comme les centrales nucléaires ou thermiques et les parcs d'éoliennes et photovoltaïques avec les consommateurs d'énergie (les maisons individuelles, les usines, …) [Farhangi, 2010] (cf. figure 4.7). L'énergie transite donc dans une seule direction : des producteurs vers les consommateurs. L'énergie produite doit être à tout moment supérieure ou égale à l'énergie consommée afin d'éviter les coupures générales de courant (blackouts). Cet équilibre entre consommation et production est, à l'heure actuelle, atteint principalement de deux manières : soit par prédiction surévaluée du besoin en se basant sur l'historique de consommation, soit par ajustement permanent de la production. Le déploiement des énergies renouvelables rend la deuxième méthode de gestion des PowerGrids difficile dans la mesure où la plupart de ces énergies ne sont pas prédictibles et encore moins adaptées à de fortes demandes ponctuelles.

Les SmartGrids sont des PowerGrids intelligents se basant sur les nouvelles technologies de l'information et de la communication [Farhangi, 2010]. Ils permettent d'assurer la meilleure balance possible entre demande et besoin en énergie en fournissant un approvisionnement sûr, durable et compétitif. Dans ce type de réseau, l'électricité circule dans les deux sens : non seulement des producteurs aux consommateurs, mais aussi des consommateurs aux producteurs.

Les MicroGrids sont une forme de SmartGrids de plus petite envergure [Ec, 2006]. Ces grilles peuvent être constituées par un ou plusieurs bâtiments avec la capacité de produire de l'électricité. Un MicroGrid est relié au SmartGrid global, mais peut fonctionner indépendamment au moyen de ses propres sources d'énergie.

Pour assurer une bonne performance pour l'utilisateur, un MicroGrid a besoin d'une certaine quantité d'informations afin de prendre des décisions appropriées [Katiraei et al., 2008]. Ces informations peuvent être brutes (comme

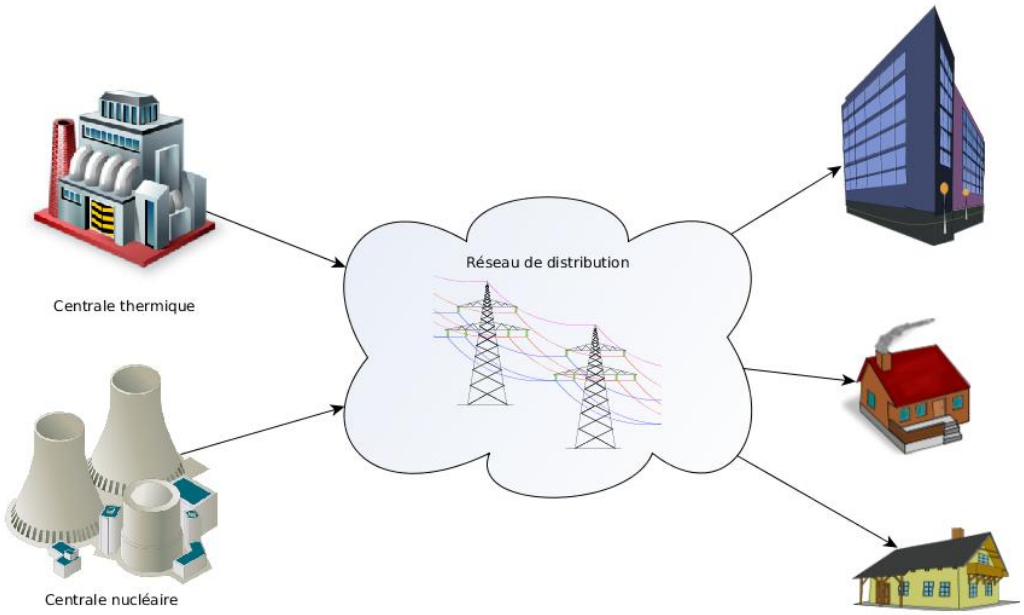


FIGURE 4.7 – Architecture d'un PowerGrid

le prix de production de l'unité, la capacité des fournisseurs ou les besoins du quartier) ou plus complexes (comme les préférences des consommateurs). Toutes ces informations créent ce qu'est appelé un plan d'informations. L'un des problèmes encore à l'étude est de savoir comment contrôler et gérer ce flux afin d'améliorer les capacités temps-réel des applications du MicroGrid. Même s'il a été démontré que l'approche décentralisée est mieux adaptée pour les MicroGrids [Abdallah et al., 2006], plusieurs propositions intéressantes se basent sur une approche centralisée [Katiraei et al., 2008].

Nous proposons d'adapter notre plateforme pour le contexte des MicroGrids afin d'offrir un mécanisme de gestion de connaissances efficace et facilement déployable en utilisant notre plan de connaissance (cf. figure 4.8). En effet, l'idée consiste à installer, dans chaque bâtiment, un seul dispositif intelligent (DI) branché à la fois aux réseaux informatiques et aux réseaux électriques. Pour éviter la complexité du réseau physique (le nombre de technologies d'accès au support (Wifi, CPL et LTE) et le nombre de fournisseurs de réseau) et améliorer la résilience du système, ces DIs sont interconnectées à travers un réseau recouvrant.

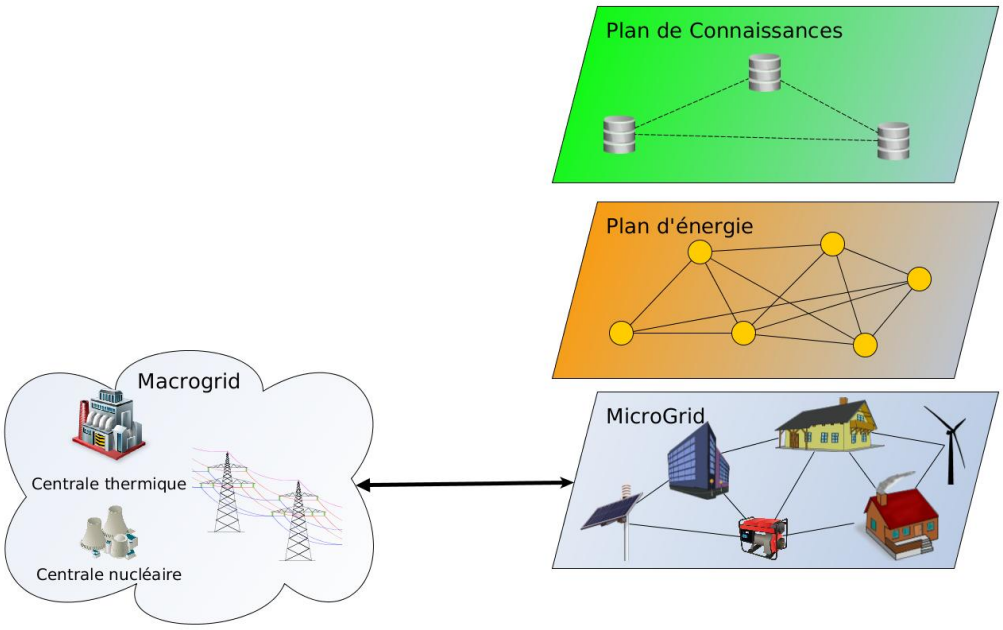


FIGURE 4.8 – Intégration du plan de connaissance dans la gestion du Micro-Grid

4.3.2 Les MicroGrids

Un MicroGrid est un sous-réseau connecté à un réseau global d'énergie [Ec, 2006], mais conçu pour fonctionner de manière autonome, en gérant intelligemment ses dépenses et ses capacités de production pour assurer le niveau "adapté" de service. Son caractère localisé lui permet de répondre efficacement et avec précision aux besoins en énergie, et garantit des niveaux adéquats de qualité, de sécurité, de fiabilité et de disponibilité. Pour atteindre cette quasi-indépendance, un MicroGrid a besoin de son propre système de contrôle. Un système de contrôle régule la communication entre les différents composants du MicroGrid (source de production, composante de stockage, point de consommation) et permet une meilleure intégration des énergies renouvelables intermittentes en ajustant l'offre et la demande en temps réel. Le système de contrôle assure également l'interopérabilité entre le MicroGrid et le MacroGrid.

Conçus à l'origine pour les régions isolées, où il est difficile de transmettre l'électricité, les Microgrids ont dépassé leur cadre initial pour intéresser les usines, les hôpitaux et autres grands consommateurs d'électricité. Un tel système nécessite une grande quantité d'informations pour atteindre ses objectifs.

Ces informations doivent être gérées et diffusées à travers tout le réseau.

4.3.2.1 Informations et connaissances

Fang et al. [2011] définissent les connaissances nécessaires au bon fonctionnement d'un MicroGrid comme toute information pouvant être mesurée ou collectée. Pour notre part, nous jugeons que ce genre d'informations sont nécessaires mais pas suffisantes pour permettre à des algorithmes de contrôle de prendre la meilleure décision possible. En plus des informations collectées et mesurées, un mécanisme de contrôle de MicroGrid nécessite des connaissances apprises telles que :

— Les composants du MicroGrid (producteurs, consommateurs, réseau de transit, …),
— Les caractéristiques de chaque fournisseur (fiabilité, capacité, prix, affinités avec les technologies vertes, …),
— Les caractéristiques de chaque consommateur (besoins, préférences, …),
— Les préférences de l'utilisateur (par exemple : un consommateur peut préférer payer plus cher une énergie propre, ou au contraire, préférer l'énergie la moins chère).

4.3.2.2 Mécanismes de diffusion des connaissances dans le MicroGrid

Les approches centralisées [Fang et al., 2011] basées sur un seul agrégateur qui gère et fournit les connaissances sont les plus adaptées en termes de temps de convergence. En effet, le système possède une seule base de données complète et n'a pas besoin d'un mécanisme de synchronisation. En outre, pour récupérer une information mise à jour, toute application de contrôle MicroGrid a juste à appeler l'agrégateur pour l'obtenir. Cependant, cette approche est sujette à des goulots d'étranglement et à des problèmes de tolérance aux pannes (panne de l'agrégateur ou rupture de liaison entre le contrôleur du MicroGrid et l'agrégateur). Dans [Nagothu et al., 2012], les auteurs proposent de stocker toutes les connaissances dans le Cloud. Ces mécanismes améliorent la capacité de tolérance aux pannes, mais sont inefficaces dans certains cas comme une rupture de la connexion Internet.

Les approches décentralisées proposées dans la littérature utilisent des tables de hachage dynamiques (DHT) pour la gestion des connaissances [Amoretti, 2009 ; Eger et al., 2008]. Cette méthode est conçue pour être aussi évolu-

tive que possible, générant un overhead limité. Cependant, les applications de contrôle doivent d'abord localiser l'information avant de la demander, ce qui pose problème dans le cas d'un besoin en temps réel de la gestion de l'énergie. La plateforme que nous avons développée, nommée SMILAY, se base sur une architecture hiérarchique et permet d'apporter une réponse à ce besoin.

4.3.3 Résultats de l'évaluation des performances

4.3.3.1 Scénario de simulation

Pour évaluer les performances de notre approche (notée SMILAY) dans un contexte de MicroGrid, nous avons généré une topologie de réseau qui représente un quartier résidentiel comme le montre la figure 4.9. Nous faisons varier le nombre de bâtiments MicroGrid de 50 à 1000 bâtiments. Chacun des bâtiments contient un petit dispositif intelligent (DI) qui représente une sorte de mini-ordinateur relié à la fois au réseau Internet et au réseau électrique. Nous estimons que chaque DI ne peut gérer que 10 autres DIs et le temps d'aller-retour maximal autorisé entre un DI et le DI qui le gère (son maître) est fixé à 35ms. Nous générons deux messages d'information par période d'une seconde et chaque DI du MicroGrid envoie 4 sondes à ses voisins par période d'une seconde pour maintenir la topologie overlay déployée.

Nous comparons notre approche avec :

— Une approche centralisée [Fang et al., 2011],
— Une approche basée sur un DHT-Chord [Amoretti, 2009 ; Eger et al., 2008] (le nombre de miroirs n en DHT est fixé à trois),
— Une approche gloutone de diffusion globale (chaque DI envoie périodiquement les connaissances à tous les autres),
— Un algorithme de diffusion à 1-saut (chaque DI envoie périodiquement des informations à tous ses voisins).

4.3.3.2 Résultats des simulations

L'évaluation que nous avons menée repose sur trois paramètres : le temps nécessaire pour diffuser l'information à toutes les parties du MicroGrid, le temps de recherche (ou latence) et l'overhead généré par chaque DI. Le temps de recherche d'une information montre la capacité de l'ensemble MicroGrid de réagir rapidement et d'adapter ainsi sa décision de la demande d'énergie.

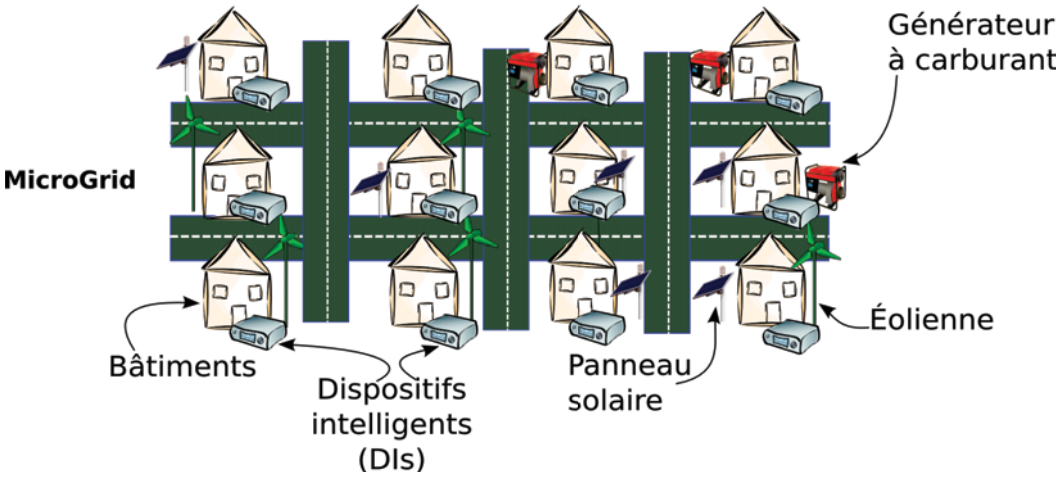


FIGURE 4.9 – Exemple d'architecture d'un MicroGrid

Le temps de dissémination influe sur la consistance de l'information et l'exactitude des réactions du contrôleur. L'overhead est corrélé à la surcharge du système.

4.3.3.2.1 Temps de dissémination Étant donné le caractère temps-réel de la gestion du MicroGrid, le temps nécessaire pour partager une connaissance sur le réseau est un paramètre critique. En effet, plus il est court, plus la prise de décision des applications de contrôle est appropriée. La figure 4.10 indique la vitesse (en millisecondes) nécessaire à chaque algorithme pour la convergence de la base de données.

Mis à part l'approche centralisée (où chaque DI envoie périodiquement des informations à un agrégateur unique), le mécanisme de diffusion globale représente la meilleure façon d'assurer une dissémination rapide des informations et ceci indépendamment de la taille du MicroGrid. L'approche DHT, comme toutes les approches réactives, estime que l'information est diffusée une fois qu'elle est reproduite à n noeuds dans le réseau. Ceci explique les bons résultats présentés dans la figure 4.10. Notre approche, appelée SMILAY, reste plus lente que le mécanisme de diffusion globale, mais reste acceptable. En effet, ce mécanisme nécessite une période d'une seconde (cas de 50 DIs) à 1,3 seconde (cas de 1000 DIs) pour diffuser des informations dans toute les parties du réseau. Le mécanisme de diffusion à 1-saut donne les plus mauvaises performances en termes de temps de diffusion. On peut noter ici qu'il augmente avec le nombre de DIs dans le réseau et atteint 5,70 secondes dans le cas de 1000 DIs, représentant ainsi une durée 4 fois plus longue que SMILAY et 80

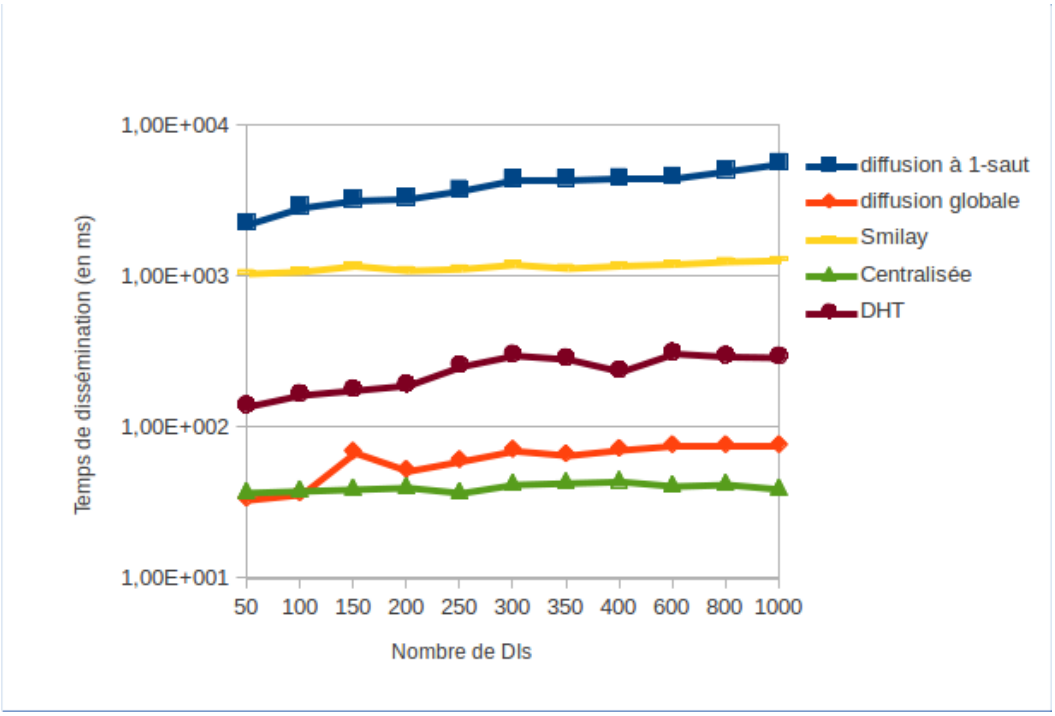


FIGURE 4.10 – Vitesse de convergence d'une information

fois celle obtenue par la méthode de diffusion globale.

4.3.3.2.2 Overhead généré L'overhead généré par les algorithmes de diffusion de l'information a un impact direct sur la surcharge du système. En effet, si celui-ci est très élevé, il peut conduire à un dysfonctionnement du réseau (un nombre important de paquets de partage de connaissances et de contrôle peut conduire à la congestion du système) et augmenter ainsi le coût de fonctionnement et la consommation énergétique. La figure 4.11 montre l'overhead généré par les algorithmes de diffusion (en kbits/s).

La supériorité des approches centralisées en termes d'overhead est évidente. En effet, chaque DI envoie ses connaissances à un seul agrégateur. Dans l'approche SMILAY, le même comportement est observé en raison de l'architecture hiérarchique. Dans la diffusion à 1-saut, chaque DI envoie des informations uniquement à ses voisins, et le nombre de voisins étant limité. Dans l'approche DHT, chaque noeud réplique les connaissances uniquement sur n noeuds. L'overhead généré par les méthodes diffusion à 1-saut, DHT ou SMILAY est toujours constant et proche de la performance de l'approche centralisée, indépendamment du nombre de DIs dans le réseau. Le mécanisme de

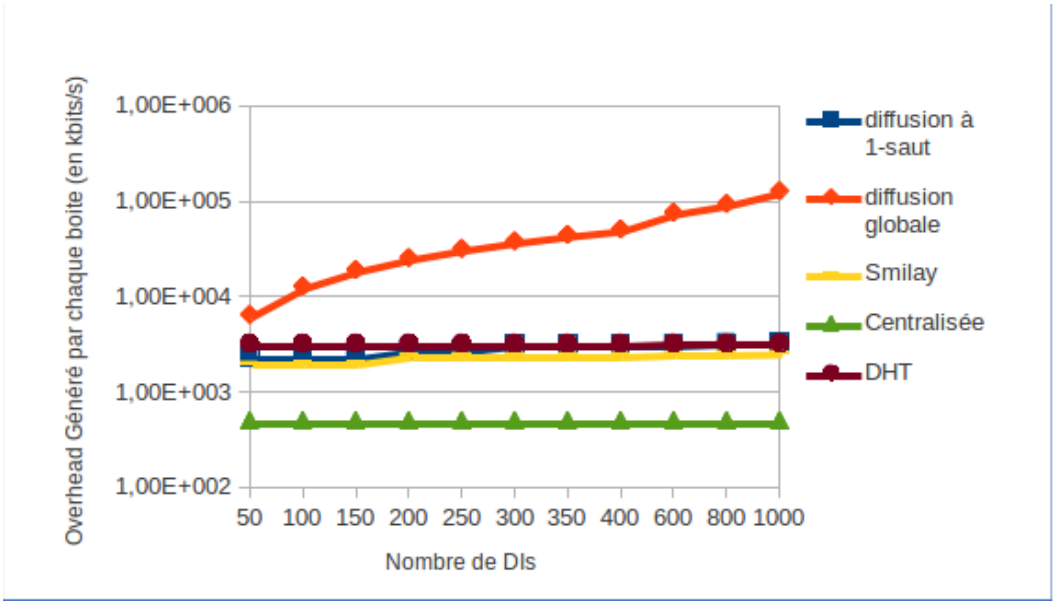


FIGURE 4.11 – overhead généré par chaque DI

diffusion globale n'est pas du tout réaliste. Il est en effet 40 fois plus mauvais que la diffusion à 1-saut et DHT et 50 fois plus mauvais que SMILAY.

4.3.3.2.3 La latence Dans les systèmes temps réel, la latence est un paramètre important. Une latence élevée peut conduire à des décisions inappropriées. La figure 4.12 montre le temps de recherche des connaissances (en ms) pour chacun des algorithmes étudiés. Nous ne considérons que l'approche centralisée, l'approche DHT et le SMILAY dans cette figure.

Dans l'approche centralisée, chaque DI demande des renseignements à l'agrégateur tandis que dans SMILAY, chaque DI les demande à son super noeud. Dans l'algorithme DHT, la demande se fait dans un temps logarithmique. Le temps de recherche requis par l'approche centralisée ou l'approche SMILAY reste constant indépendamment du nombre de DIs dans le réseau. Il s'avère aussi que le temps de recherche nécessaire à l'approche DHT augmente avec le nombre de noeuds et dépasse 150 fois le temps de recherche nécessaire à SMILAY.

Les figures 4.10, 4.11 et 4.12 montrent que SMILAY offre le meilleur compromis sur les 5 approches étudiées. En effet, l'overhead généré reste contenu tout en offrant de bons résultats en termes de temps de convergence. De plus, SMILAY obtient de meilleurs résultats comparativement aux autres approches

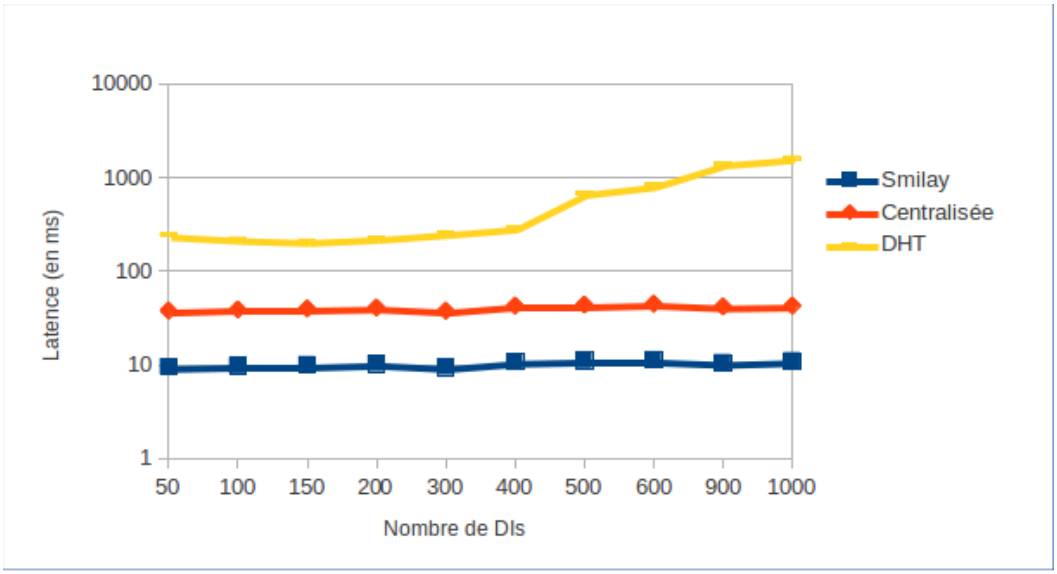


FIGURE 4.12 – Temps de recherche(en ms)

pour les temps de latence.

4.3.4 Conclusion

Ce chapitre est dédié à des validations expérimentales de notre plateforme de gestion des connaissances dans le cadre de deux exemples d'utilisation.

Au moment où nous assistons à une multitude d'offres, le premier cas concerne la problématique de la sélection d'un fournisseur Cloud par une entreprise tierce au regard de ressources réseaux auxquelles elle souhaite s'abonner. Nous avons proposé deux nouvelles approches pour répondre à ce besoin. Pour cela, nous avons fixé et mesuré les indicateurs clés de performance. Ces données alimentent la base de connaissances de chaque contrôleur, disséminées ensuite à l'ensemble des contrôleurs au travers de notre plateforme. L'ensemble de ces bases forme le plan global de connaissance. Nous avons ensuite implémenté un algorithme de sélection de fournisseurs (SCPS) basé sur une approche de type programmation linéaire. Au regard du caractère dynamique du réseau, l'approche proposée s'est avérée insuffisante. Nous avons été amenés à y intégrer une technique sur la base de l'algorithme de Page-Hinkley. La nouvelle approche résultante permet ainsi de choisir de manière évolutive le fournisseur Cloud le plus adapté aux besoins du client et ceci en tout instant. L'expérimentation réalisée montre que la méthode adaptative (ASPS) offre

des résultats meilleurs que la méthode simple (SCPS).

Dans le deuxième cas, nous nous sommes intéressés à un autre exemple, celui de la gestion des connaissances dans le cas des MicroGrids qui représentent des nouveaux types de SmartGrids. Il s'agit de petits SmartGrids autogérés et capables de prendre les décisions appropriées pour assurer à l'utilisateur une gestion optimisée de l'énergie consommée. Pour y parvenir, nous mettons en oeuvre un plan de connaissance déployé dans un MicroGrid afin de rendre celui-ci capable de gérer et de partager toutes les informations nécessaires à l'ensemble de ses composants. Nous avons démontré que notre approche peut être facilement adaptée à ce nouveau cadre. Cette variante, appelée SMILAY, est basée sur le déploiement d'un ensemble de dispositifs intelligents (DIs) dans le MicroGrid. Certains de ces DIs sont sélectionnés pour la gestion de l'information nécessaire aux algorithmes de contrôle du MicroGrid. Les résultats de l'évaluation des performances ont clairement démontré l'efficacité du modèle proposé. En effet, dans nos conditions de simulation, la plateforme SMILAY obtient de meilleures performances comparativement aux autres approches distribuées en termes de temps de diffusion de l'information, de surcharge (overhead) et de temps de récupération de ces informations.

Conclusion

Cette thèse avait pour objectif de proposer une approche basée sur les connaissances pour l'intégration de l'autonomique dans les réseaux de communication et de démontrer sa pertinence au travers de son utilisation dans un certain nombre de cas d'usage.

Le concept de l'autonomique représente un nouveau paradigme qui a pour but de rendre le réseau indépendant de toute gestion humaine en le dotant de capacités "d'auto-*" (auto-configuration, auto-réparation, auto-optimisation et auto-protection) ; un des objectifs majeurs est celui de raccourcir les délais de réaction des différents contrôles opérés dans le réseau au regard de la dynamicité de celui-ci.

Pour garantir les fonctionnalités liées à l'autonomique, il est nécessaire de construire un ensemble de connaissances, par apprentissage ou par saisie, portant sur des informations diverses liées au fonctionnement du réseau : Paramètres techniques, Perception des usagers, Historique des interventions effectuées sur le réseau, etc. Ces connaissances doivent être agrégées et gérées au sein d'un nouveau plan appelé "plan de connaissance" qui vient se rajouter aux 3 plans traditionnels existants : Données, Contrôle et Gestion.

L'émergence d'un tel plan induit de travailler sur plusieurs aspects liés à la gestion de cette connaissnce : sa définition, sa construction, sa représentation ou encore sa dissémination. Dans le cadre de cette thèse, nous nous sommes intéressés à la conception, à l'implémentation et à la mise en application d'une plateforme générique de gestion de connaissances sous licence libre, nommé Autolay, répondant à un certain ensemble de contraintes.

Dans ce contexte, plusieurs approches de dissémination de connaissances existent et utilisent différentes approches que nous avons classées en quatres catégories : Vues situées [Nguengang et al., 2008 ; Bullot et al., 2008 ; Marrow and Manzalini, 2006], Tables de hashage distribuées (DHT) [Strassner et al., 2006 ; Li et al., 2008], Usage du Cloud [Yu et al., 2012], et Diffusion locale [Tang et al., 2007 ; Schuetz et al., 2007], …). Tout en étant intéressantes sur certains aspects, ces approches ne répondent pas aux objectifs que nous nous sommes fixés. Ces derniers portent sur les aspects liés à un déploiement à grande échelle, une tolérance aux pannes et une vision globale des connaissances. Les travaux développés ici nous ont conduit à proposer une nouvelle approche de dissémination des connaissances qui se base sur deux idées clés : i) une

architecture hiérarchique constituée de super noeuds et d'hyper noeuds pour assurer un déploiement à grande échelle tout en gardant une vision globale des connaissances, ii) la construction de plusieurs instances de ce plan de connaissance ; chaque instance est amenée à être activée de manière adaptative et corrélée au contexte d'utilisation du réseau. Un tel contexte dépend de plusieurs paramètres que nous étions amenés à définir et à modéliser.

Afin de construire notre architecture hiérarchique, nous avons, dans un premier temps, étudié les approches existantes de sélection des super noeuds. Cette étude a montré qu'elles sont opérationnelles pour un usage unique et pour une problématique spécifique (réseau de capteurs sans fil, réseau pair-à-pair de partage de fichiers, …). Aucune ne répond au souci d'un usage multiple pour lequel des contraintes spécifiques concernant à la fois la minimisation du délai de propagation des connaissances et l'équilibrage de charges sont primordiales. Nous avons donc proposé une première approche, nommée DCLARA, qui s'appuie sur un partitionnement de type k-medoïd. Cette méthode consiste à découper le réseau en zones et à sélectionner un noeud (appelé hyper noeud) dans chaque zone. L'ensemble des hyper noeuds se partage les connaissances sur la topologie et les paramètres du réseau de manière à ce que chacun d'entre eux puisse dérouler un algorithme de partitionnement global (CLARA [Soni and Ganatra, 2012]) et répercuter ensuite le résultat obtenu sur sa propre zone.

Dans cette première proposition, le choix des super noeuds est fait de manière permanente. L'étude menée dans le cas de réseaux fortement dynamique a montré que cette manière de choisir les super noeuds conduit à une dégradation progressive du mécanisme de diffusion des connaissances (augmentation de la latence lors de la récupération des connaissances et du temps de leur dissémination). Pour éviter cette dégradation, nous avons étendu notre première proposition en lui rajoutant une amélioration, nommée, DCLARA-PH, qui se base sur le test statistique de Page-Hinkley. Cette nouvelle contribution permet de procéder, sans aucune hypothèse préalable, à une reconfiguration de l'architecture hiérarchique courante et donc à la reconstruction d'un nouvel ensemble de super noeuds.

Notre troisième contribution consiste à proposer un mécanisme permettant de relier les noeuds ainsi sélectionnés par un ensemble de réseaux recouvrants (Overlay). Chacun d'entre eux est en charge d'une instance du plan de connaissance. Ces instances sont construites sur la base de critères fixés a priori par l'utilisateur et pouvant répondre à différents types d'usage : importance de la connaissance, sa position géographique, type d'application souhaitée, etc. Le plan de connaissance actif sera ainsi instancié sur la base de ces repré-

sentations (par reconfiguration automatique) prenant en compte les critères sélectionnés.

Ces 3 propositions ont constitué le socle sur lequel nous avons conçu et construit notre plateforme de gestion de connaissances. Dans cette plateforme, les connaissances sont gérées par un système de gestion de bases de données orienté document (de type NoSQL). Un tel choix s'explique par la flexibilité dans la représentation des connaissances qu'une telle structure offre ou encore la possibilité de la conservation de l'historique des mises à jour d'une connaissance donnée.

A des fins de validation, l'approche proposée a été comparée à trois catégories d'approches parmis celles utilisées au sein de la communauté : celles utilisant des tables de hachage distribuées (DHT-Chord) [Stoica et al., 2001], celles basées sur la diffusion (broadcast) et celles sélectionnant aléatoirement les super noeuds [Abdeljaouad and Karmouch, 2012]. Les résultats obtenus dans cette étude expérimentale ont montré l'efficacité de nos propositions (DCLARA et DCLARA-PH) en termes de temps de dissémination des connaissances, de latence lors de la récupération des connaissances et de surcharge générée par les protocoles proposés.

Une autre étude a porté sur l'usage de la plateforme développée dans un environnement virtualisé. Nous avons proposé à cet effet une méthode consistant à instancier le plan de connaissance global en plusieurs représentations différentes (appelées sous plan) et ensuite à construire le meilleur réseau recouvrant sur lequel sera déployée la représentation sélectionnée sur la base de critères fixés a priori. Nous avons pour cela classer les connaissances en 3 catégories : connaissances de basse priorité, connaissances de moyenne priorité et connaissances de haute priorité. Ce degré de priorité est fixé en fonction de l'impact de la connaissance sur le fonctionnement du réseau. Ainsi, une panne sur un lien ou sur un noeud est considérée comme connaissance de haute priorité, les autres paramètres du réseau (bande passante résiduelle, latence, ...) sont considérés comme des connaissances à priorité moyenne. Les informations non vitales tels que l'actualisation du contenu d'un serveur de vidéo à la demande sont traitées comme étant de basse priorité.

Dans un cadre lié à l'utilisation de nos approches dans les réseaux logiciels, plus connus sous l'appellation anglo-saxone *Software Defined Networks* (SDN) et qui ont connu un fort développement ces 5 dernières années tant du point de vue de la recherche académique que dans le secteur industriel, nous avons appliqué notre plateforme à deux cas d'usage que nous avons sélectionnés sur la base de leur acuité actuelle. Le premier cas concerne la reconfigura-

tion dynamique des réseaux recouvrants pour une application de diffusion de contenus, largement répandue chez les opérateurs de délivrance de contenus (TelCo Content Delivery Operators). L'autre cas est lié à la sécurisation d'un réseau opérateur et plus spécialement dans le cas d'une attaque de type déni de service distribué. Notre objectif ici a été de tester l'apport de l'approche développée pour la dissémination des connaissances dans ce type de scénario et de regarder de plus près les temps de réaction du réseau pour la mise en route d'actions de correction comparativement aux approches classiques.

Comme notre plateforme se veut à usages multiples, elle offre un cadre générique pouvant être utilisé dans plusieurs contextes comme par exemple les réseaux de diffusion de contenu (CDNs), les réseaux centrés sur l'information (ICNs), le Cloud Computing ou encore les SmartGrids. Dans le chapitre 4, nous décrivons deux cas d'utilisation, que nous avons sélectionnés pour leur diversité, afin d'en illustrer l'usage. Dans le premier cas, notre intérêt s'est porté sur le domaine grandissant de l'usage du Cloud Computing et de la multitude d'offres dans ce domaine. Nous y avons traité le problème de la sélection adaptative des opérateurs Cloud au regard de contraintes dynamiques et évolutives fixées par l'utilisateur. Le second cas concerne un tout autre sujet, tout aussi d'actualité ; il s'agit du cas des SmartGrids. Nous y traitons plus particulièrement la propagation des informations dans un système d'information nécessaire au transport de l'électricité dans des MicroGrids. Dans chacun des cas traités, nous avons montré l'adaptabilité de la plateforme développée et l'apport du plan de connaissance dans le fonctionnement des systèmes sous jacents.

À l'issue de ce travail, nous dégageons plusieurs perspectives de recherche que nous évoquons ci-dessous :

— L'interopérabilité de la représentation des connaissances représente une piste à investiguer. En effet, plusieurs modèles de représentation des connaissances dans le réseau ont été proposés tels que la MIB (Management Information Base) [McCloghrie and Rose, 1991], CIM (Common Information Model) [DMTF, 1999], NDL (network Description Langauge) [Ham, 2010] sans pouvoir aboutir à un standard unique. Dans ce contexte, il serait intéressant que la plateforme développée dans cette thèse intègre une interface afin d'assurer une interopérabilité avec les modèles de représentation utilisés dans les autres plateformes existantes. Pour permettre cela, des méthodes d'alignement d'ontologies (qui consistent à retrouver les équivalences entres les différents concepts définis dans les différentes représentations) sont en effet à investiguer.

— La mise à jour en continu du plan de connaissance et son alimentation en nouvelles connaissances constitue un facteur crucial pour le fonctionnement global du système. Des méthodes d'apprentissage, inductives ou déductives, sont à étudier dans ce cadre. Elles permettent, par exemple, une classification automatisée des connaissances en termes de priorités de dissémination, ou encore, l'anticipation et la prédiction des dysfonctionnements du système, par la corrélation de certaines connaissances qui pourraient apparaître comme distinctes.
— Nous avons présenté, dans cette thèse, deux cas d'usage complètement dissociés dans le cadre des réseaux logiciels (SDN). Ces exemples avaient pour but de montrer que la plateforme que nous avons proposée peut être une alternative décentralisée et réelle pour le contrôle de ces réseaux. Toutefois, plusieurs aspects restent à étudier, notamment ceux relatifs à la distribution des fonctions de contrôle sur les super noeuds ou encore ceux liés aux algorithmes de contrôle eux mêmes comme les mécanismes d'anticipation, de détection, de diagnostic ou de réparation de pannes.
— Sur un tout autre chapitre, celui du point de vue fonctionnel, les MicroGrids (et par extension les SmartGrids) peuvent être divisés en 3 plans : un plan d'information, un plan de gestion et un plan de protection. Dans cette thèse, nous avons montré que notre plateforme peut être adaptée pour servir de plan d'information pour les MicroGrids. Afin de satisfaire au mieux l'utilisateur final au regard du service fourni, il serait intéressant d'étudier, dans la continuité de ce travail, l'intégration de l'autonomique dans la gestion des MicroGrids dans le but d'améliorer leur fonctionnement (par exemple, piloter dynamiquement l'approvisionnement et la production d'énergie en fonction de la consomation)
— Dans le cadre de cette thèse, nous avons été amenés à faire un choix sur un ensemble de paramètres d'évaluation. Le premier concerne la définition d'une zone. Ce paramètre peut avoir un impact non négligeable sur les performances (vitesse de stabilisation) de notre architecture. Ainsi, il serait intéressant de pouvoir le définir dynamiquement et en fonction du contexte (réseau d'opérateur, réseau de capteur, réseau de domicile, …). Le second concerne la capacité des super nœuds. Dans un usage réel et multiple, cette métrique doit pourvoir évoluer afin de s'adapter au contexte d'exploitation de la plateforme. Ainsi, dans le cadre d'un contrôle distribué des réseaux logiciels, un super nœud doit pouvoir, en plus de gérer les connaissances de sa zone, assurer le bon fonctionnement d'un ensemble d'équipements réseau à sa charge. Dans le cadre des réseaux de contenus, les super noeuds peuvent jouer le rôle d'un

serveur de cache, ce qui nécessite beaucoup plus de ressources surtout si les données en question sont de nature multimédia.

— À long terme, nous pensons que l'intégration de l'autonomique dans les différents constituants du réseau est amenée à rendre les réseaux beaucoup plus réactifs et à diminuer les coûts d'exploitation des opérateurs. Cette intégration permettra à un opérateur de fournir un bien meilleur service à ses utilisateurs. En effet, la vision de l'administration du réseau évoluera vers une méta-administration où l'opérateur n'aura qu'a définir des objectifs de haut niveau fixés sur la base de la perception des usagers (cf. figure 4.13). Dans ce réseau, l'administrateur pourra agir sur les mécanismes de gestion et de contrôle de l'ensemble des équipements du réseau : du serveur d'application jusqu'à la borne d'accès au réseau de l'utilisateur final.

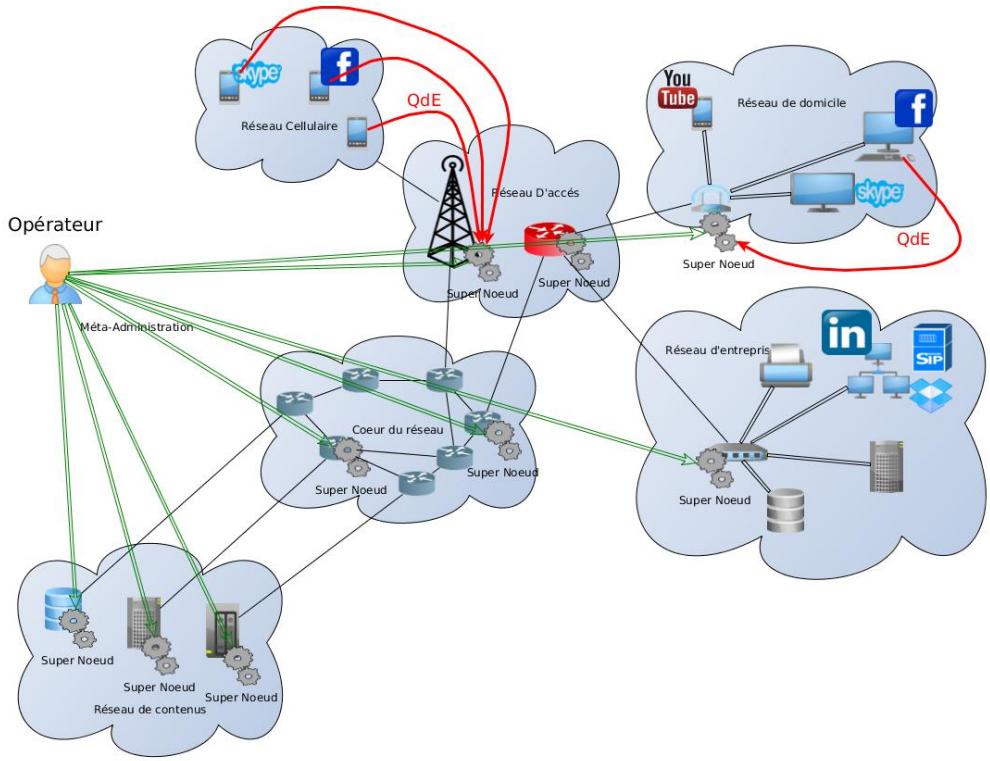


FIGURE 4.13 – Vision à long terme : Méta-administration

Annexe A

NExTLab

Introduction

Nous avons été amenés à développer la plateforme d'expérimentation réseau NExTLab (**N**etwork **Ex**perimental **T**ools for research **Lab**oratory) afin d'évaluer notre plateforme de gestion de connaissances. NExTLab permet la création de topologies pour des réseaux virtuels peuplés de noeuds virtuels exécutant des logiciels standards. Elle est mise à la disposition de l'ensemble de la communauté. À la différence des outils d'expérimentation du réseau comme VIRCONEL [Benchaïb and Hecker, 2008 ; Benchaib and Hecker, 2011] ou CORE [Ahrenholz et al., 2008], NExTLab permet de gérer plusieurs utilisateurs, plusieurs expériences simultanées ainsi que plusieurs types d'hôtes. La plateforme qui se rapproche le plus de NExTLab est ToMaTo [Schwerdel et al., 2012]. Toutefois, la politique de déploiement de nouvelles fonctionnalités est plus aisée dans NExTLab.

A.1 Objectif

L'objectif de NExTLab est de permettre à tout utilisateur de créer et d'utiliser des topologies de réseaux pour leurs expérimentations (voir figure A.1). Une topologie est constituée de deux types de composants : les noeuds (tels que les routeurs, les commutateurs et les ordinateurs) et les liens.

A.2 Architecture

L'architecture de NExTLab s'articule autour de 2 modules : i) le serveur de déploiement et ii) l'interface de gestion.

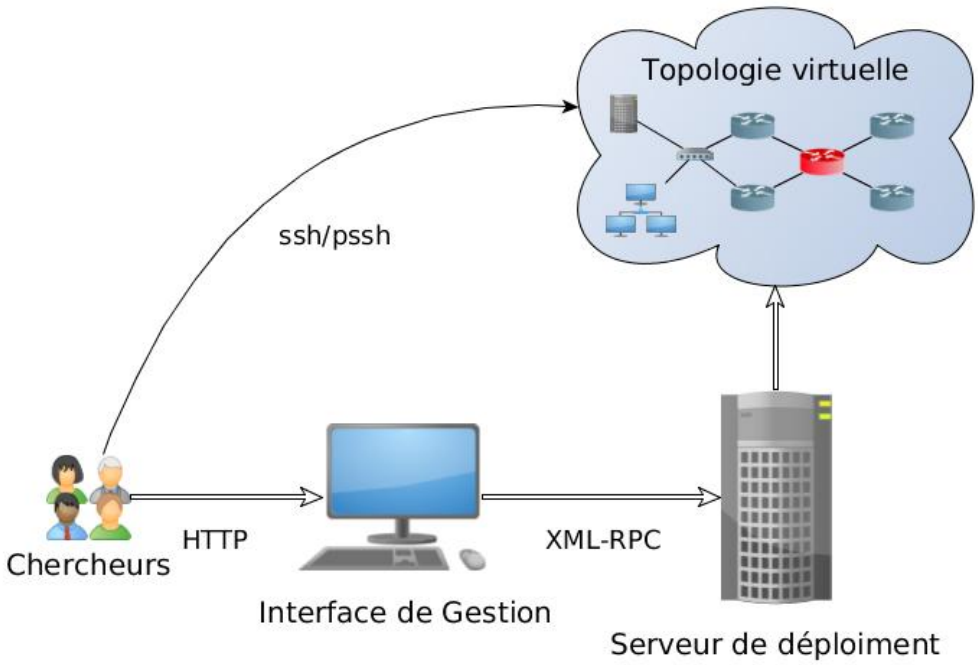


FIGURE A.1 – Architecture de NExtLab

Le serveur de déploiment exécute l'environnement de virtualisation libre "Proxmox Virtual Environment" (PVE) [1]. Cet environnement se présente comme un hyperviseur sous licence AGPLv3. PVE supporte deux types de virtualisation :

— Une virtualisation complète (KVM) qui permet la virtualisation de tout système d'exploitation (y compris des systèmes Microsoft et BSD). Elle offre la possibilité de faire tourner un noyau sur chaque machine virtuelle et d'émuler l'ensemble des périphériques de la machine physique.
— Une virtualisation OpenVZ (ou par conteneur) qui permet la création d'instances de systèmes d'exploitation isolées qui s'exécutent sur le même noyau, appelées "Conteneurs" . Cette virtualisation est très performante ; elle consomme en effet très peu de ressources. En contrepartie, il n'est pas possible de faire des modifications du noyau. De plus, il n'est possible que de créer des instances GNU/Linux.

PVE propose une API [2] qui peut être utilisée pour la création d'interfaces frontales. Nous en avons réalisé une sous la forme d'une application web. Cette

1. http ://pve.proxmox.com
2. Application Programming Interface

application permet aux utilisateurs de créer et d'éditer leurs topologies. Deux types d'éditeurs ont été développés :

i) un éditeur graphique, représenté par la figure A.2, où l'utilisateur peut facilement visualiser la topologie finale créée.

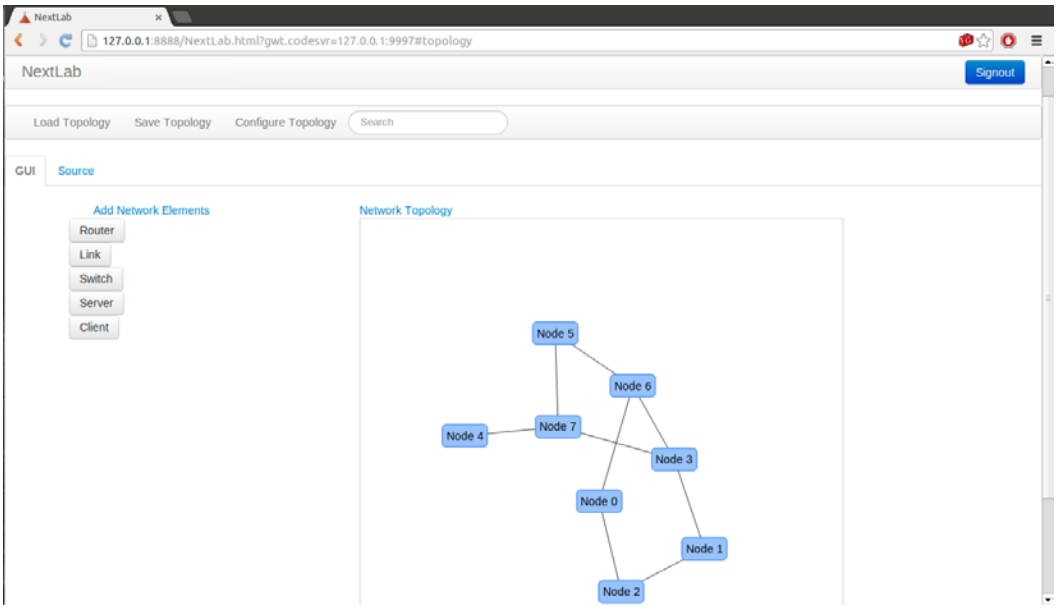


FIGURE A.2 – Éditeur graphique

ii) un éditeur textuel, représenté par la figure A.3, basé sur le format JSON (JavaScript Object Notation) qui permet à l'utilisateur de personnaliser les éléments de la topologie (par exemple, spécifier les caractéristiques d'un lien ou le type d'un routeur). Il permet aussi de configurer le plan d'adressage IP ou déléguer cela au configurateur automatique de NExT-Lab.

L'utilisateur a la possibilité de sauvegarder une topologie, charger une ancienne afin de la compléter ou encore supprimer une existante.

A.3 Éléments de NExTLab

La plateforme NExTLab permet à l'utilisateur de mettre en place une topologie à partir des composants suivants :

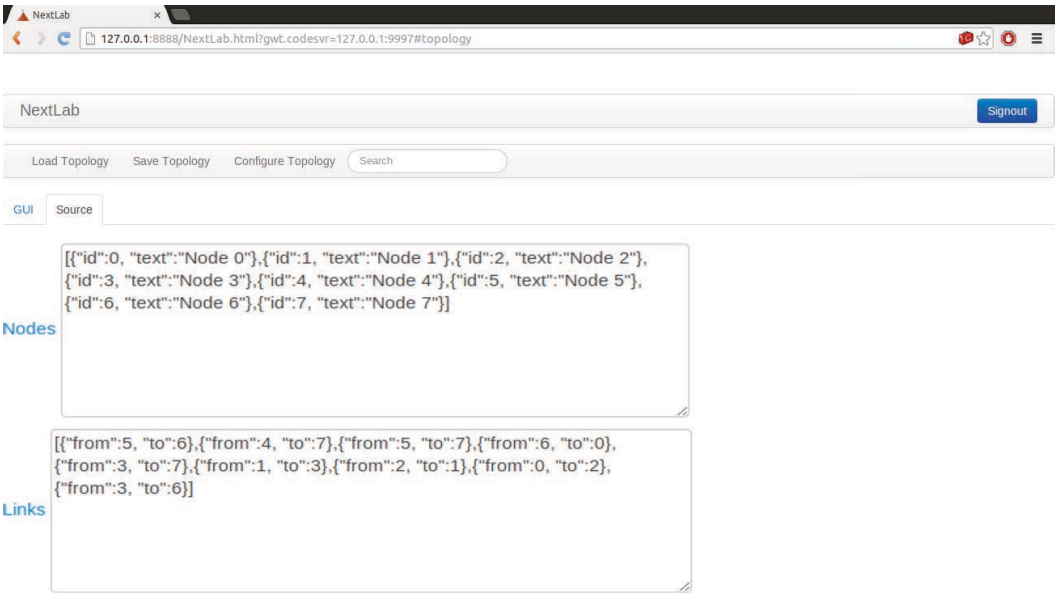


FIGURE A.3 – Éditeur textuel

Un Routeur est une machine virtuelle linux ayant des fonctions de routage. L'utilisateur peut personnaliser son routeur en y ajoutant des fonctionnalités (comme c'est le cas de notre plateforme de gestion de connaissances). Ces fonctionnalités sont téléchargeables depuis un répertoire partagé entre tous les routeurs instanciés et activables en les ajoutant comme une routine au démarrage du noeud.

Un switch ou commutateur est émulé par un réseau local virtuel (VLAN).

Un lien est un commutateur ayant uniquement deux interfaces. Toutefois, à la différence d'un commutateur, les caractéristiques d'un lien sont gérables via l'émulateur réseau NetEm (Network Emulator)[3]. Il s'agit d'un émulateur réseau pouvant gérer plusieurs métriques : le délai, la gigue, le taux de perte, la duplication des paquets et leur ordre d'arrivée. Il est disponible à partir de la version 2.6 du noyau Linux.

Un Client est une machine virtuelle Linux ayant, par défaut, certaines fonctionnalités telles qu'un lecteur vidéo ou un navigateur fonctionnant en mode console.

Un Serveur est une machine virtuelle Linux ayant certaines fonctionnalités préinstallées comme un serveur de diffusion de vidéos ou un serveur web.

3. http ://www.linuxfoundation.org/collaborate/workgroups/networking/netem

Par souci de passage à l'échelle, NExTLab se contente d'exploiter la virtualisation par conteneurs pour l'émulation des noeuds du réseau.

A.4 Gestion des utilisateurs dans NExTLab

L'objectif de NExTLab est de gérer un ensemble d'expérimentations réseau. En effet, outre le module de gestion des utilisateurs (ajout et retrait d'un utilisateur) et de leurs droits (administrateur ou simple utilisateur), un ordonnanceur de tâches a été développé.

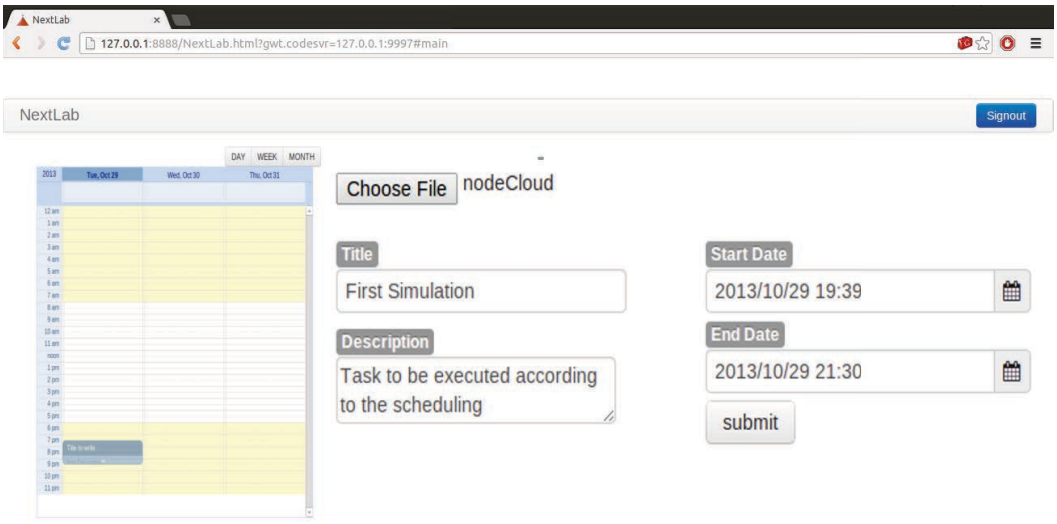


FIGURE A.4 – Ordonnanceur de tâches

Comme le montre la figure A.4, cet ordonnanceur permet à un utilisateur connecté de :

— Visualiser les tâches planifiées sur l'émulateur, leurs descriptions, ainsi que l'utilisateur qui a planifié leurs exécutions.
— Ajouter une nouvelle tâche avec une topologie déjà créée.
— Différer l'exécution d'une tâche planifiée.

En fonction de la taille de la topologie instanciée par chaque tâche, NExTLab détermine s'il est possible de lancer plusieurs tâches en parallèle.

A.5 Conclusion

NExTLab offre aux utilisateurs la possibilité d'expérimenter leurs différents protocoles et propositions sur des topologies de réseaux virtualisées.

La topologie la plus large que nous avons testée sur NExTLab est de 55 noeuds et 74 liens. Il est facile d'obtenir des topologies de plus grande envergure à condition que le serveur de déploiement puisse les supporter. D'une conception modulaire, NExTLab peut facilement être étendu.

Bibliographie

A. Zafeiropoulos A. Liakopoulos. Autonomic monitoring and resource management using p2p techniques. In *Terena Networking Conference*, 2009. (Cit en pages 11 et 36.)

T. Abdallah, R.A. Ducey, C.A. Feickert, United States. Army. Corps of Engineers, Engineer Research, Development Center (U.S.), and Construction Engineering Research Laboratory (U.S.). *Control Dynamics of Adaptive and Scalable Power and Energy Systems for Military Micro Grids*. US Army Corps of Engineers, Engineer Research and Development Center, Construction Engineering Research Laboratory, 2006. (Cit en page 128.)

Imad Abdeljaouad and Ahmed Karmouch. Self-organizing overlay networks for autonomic manager selection. In *Communications (ICC), 2012 IEEE International Conference on*, pages 6437–6441. IEEE, 2012. (Cit en pages 54, 80, 92 et 139.)

M. Abid, Andreas Berl, Z. Boudjemil, Abderhaman Cheniour, Steven Davy, Spyros Denazis, Alex Galis, Jean-Patrick Gelas, Claire Fahy, Andreas Fischer, Javier Rubio Loyola, Laurent Lefèvre, Daniel Macedo, Lefteris Mamatas, Hermann de Meer, Z. Movahedi, John Strassner, and H. Zimmermann. Autonomic internet initial framework - deliverable d6.1. Technical report, INRIA, 2008. (Cit en pages 37 et 38.)

Aicha Aguezzoul, Pierre Ladet, et al. Sélection et évaluation des fournisseurs : Critères et méthodes. *Revue Française de Gestion Industrielle*, 2006. (Cit en page 116.)

Jeff Ahrenholz, Claudiu Danilov, Thomas R Henderson, and Jae H Kim. Core : A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7, 2008. (Cit en page 143.)

Michele Amoretti. The peer-to-peer paradigm applied to hydrogen energy distribution. *EUROCON 2009, EUROCON'09. IEEE*, 2009. (Cit en pages 130 et 131.)

Luciano Baresi, Antonio Di Ferdinando, Antonio Manzalini, and Franco Zambonelli. The cascadas framework for autonomic communications. In *Autonomic Communication*, pages 147–168. Springer, 2009. (Cit en page 37.)

S. A. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. In *Proc. of the INFOCOM '06*, Barcelona, Spain, April 2006. (Cit en pages 58 et 59.)

Ingmar Baumgart, Bernhard Heep, and Stephan Krause. OverSim : A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, 2007. (Cit en page 91.)

Y. Benchaib and A. Hecker. Virconel : A network virtualizer. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, pages 429–432, 2011. (Cit en page 143.)

Yacine Benchaïb and Artur Hecker. Virconel : A new emulation environment for experiments with networked it systems. In *High Performance Computing & Simulation Conference*, 2008. (Cit en page 143.)

Sushil Bhardwaj, Leena Jain, and Sandeep Jain. CLOUD COMPUTING : A STUDY OF INFRASTRUCTURE AS A SERVICE (IAAS). 2, 2010. (Cit en page 113.)

Gordon S Blair, Geoff Coulson, Lynne Blair, Hector Duran-limon, Paul Grace, Rui Moreira, and Nikos Parlavantzas. Reflection, Self-Awareness and Self-Healing in OpenORB. *Analysis*, 2002. (Cit en page 29.)

Ghazi Bouabene, Christophe Jelger, Christian Tschudin, Stefan Schmid, Ariane Keller, and Martin May. The autonomic network architecture (ana). *IEEE J.Sel. A. Commun.*, 2010. (Cit en pages 37 et 38.)

Ronald J. Brachman. Systems that know what they're doing. *IEEE Intelligent Systems*, 2002. (Cit en page 42.)

Thomas Bullot, Rida Khatoun, Louis Hugues, Dominique Gaïti, and Leila Merghem-Boulahia. A situatedness-based knowledge plane for autonomic networking. *International Journal of Network Management*, 2008. (Cit en pages 49 et 137.)

Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, and A F De Rose. CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. 2011. (Cit en page 113.)

Iacopo Carreras, Imrich Chlamtac, Francesco De Pellegrini, and Daniele Miorandi. Bionets : Bio-inspired networking for pervasive communication environments. *Vehicular Technology, IEEE Transactions on*, jan. 2007. (Cit en page 36.)

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable : A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 2008. (Cit en page 50.)

Ranganai Chaparadza, Symeon Papavassiliou, Timotheos Kastrinogiannis, Martin Vigoureux, Emmanuel Dotaro, Alan Davy, Kevin Quinn, Michal Wódczak, Andras Toth, Athanassios Liakopoulos, et al. Creating a viable evolution path towards self-managing future internet via a standardizable reference model for autonomic network engineering. In *Future Internet Assembly*, pages 136–147, 2009. (Cit en page 36.)

Yan Chen, David Bindel, Han Hee Song, and Randy H. Katz. Algebra-based scalable overlay network monitoring : algorithms, evaluation, and applications. *IEEE/ACM Trans. Netw.*, 2007. (Cit en page 47.)

Cisco. Cisco visual networking index : Forecast and methodology, 2012–2017, white paper. May 2013. (Cit en page 25.)

D D Clark, C Partridge, C J Ramming, and J T Wroclawski. A knowledge plane for the internet. *in SIGCOMM '03 : Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA : ACM Press*, 2003. (Cit en pages 22, 39, 41, 44, 54 et 73.)

David Culler, Timothy Roscoe, Larry Peterson, Larry Peterson, Tom Anderson, and Tom Anderson. A blueprint for introducing disruptive technology into the internet. 2002. (Cit en pages 46, 88 et 117.)

G. Diaz and K. Chen. Distributed management to services deployment in autonomic networks. In *Information and Communication Technologies : From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6, 2008. (Cit en page 54.)

DMTF. *Common Information Model (CIM) Specification*. June 1999. (Cit en pages 48 et 140.)

Ec. Vision and Strategy for Europe's Electricity Networks of the Future. Technical report, European Commission, 2006. (Cit en pages 127 et 129.)

K. Eger, C. Gerdes, and S. Oztunali. Towards p2p technologies for the control of electrical power systems. In *Peer-to-Peer Computing , 2008. P2P '08. Eighth International Conference on*, 2008. (Cit en pages 130 et 131.)

Xi Fang, Satyajayant Misra, and Guoliang Xue. Smart Grid – The New and Improved Power Grid : A Survey. *Communications Surveys & tutorial*, 2011. (Cit en pages 130 et 131.)

H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 2010. (Cit en page 127.)

Markus Fiedler, Helmut Hlavacs, Klaus Hackbarth, and Patrik Arlos. Quality of experience. *Annals of Telecommunications*, 65(1) :1–2, 2010. (Cit en page 97.)

Razvan V. Florian. Autonomous artificial intelligent agents. Technical report, Center for Cognitive and Neural Studies (Coneural), 2003. (Cit en page 49.)

Alex Galis, Spyros G Denazis, Alessandro Bassi, Pierpaolo Giacomin, Andreas Berl, Andreas Fischer, Hermann de Meer, John Srassner, Steven Davy, Daniel F Macedo, et al. Management architecture and systems for future internet networks. In *Future Internet Assembly*, pages 112–122, 2009. (Cit en page 37.)

Pawel Garbacki, Dick H. J. Epema, and Maarten van Steen. The design and evaluation of a self-organizing superpeer network. *IEEE Trans. Comput.*, 2010. (Cit en pages 58 et 59.)

Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking : seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, New York, NY, USA, 2011. ACM. (Cit en page 26.)

Lee Gillam, Bin Li, John O¿Loughlin, and Anuz Pratap Tomar. Fair Benchmarking for Cloud Computing systems. *Journal of Cloud Computing : Advances, Systems and Applications*, 2013. (Cit en page 115.)

Vânia Gonçalves and Pieter Ballon. Adding value to the network : Mobile operators' experiments with Software-as-a-Service and Platform-as-a-Service models. *Telematics and Informatics*, 2011. (Cit en page 115.)

S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, pages 374–387, 1998. (Cit en page 57.)

J.J. Ham. *A Semantic Model for Complex Computer Networks : The Network Description Language*. Universiteit van Amsterdam, 2010. (Cit en pages 48 et 140.)

Nicholas Hatt, Axis Sivitz, and Benjamin A Kuperman. Benchmarking operating systems. Conference for Undergraduate Research in Computer Science and Mathematics, 2007. (Cit en page 121.)

Teresa W. Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of Domination in Graphs (Pure and Applied Mathematics (Marcel Dekker))*. CRC, 1998. (Cit en page 57.)

Karl Huppler. Benchmarking with Your Head in the Cloud. pages 97–110, 2012. (Cit en page 115.)

a Iosup, S Ostermann, M N Yigitbasi, R Prodan, T Fahringer, and D H J Epema. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, 2011. (Cit en page 114.)

Gian Paolo Jesi, Alberto Montresor, and Ozalp Babaoglu. Proximity-aware superpeer overlay topologies. In *Self-Managed Networks, Systems, and Services*. Springer, 2006. (Cit en pages 58 et 61.)

Isabelle Jeuge-Maynart. *Le petit Larousse illustré : en couleurs : 87 000 articles, 5 000 illustrations, 321 cartes, chronologie universelle*. Larousse, Paris, 2010. (Cit en pages 45 et 46.)

Christos Kaklamanis, Danny Krizanc, and Satish Rao. Hot-potato routing on processor arrays. In *Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, pages 273–282. ACM, 1993. (Cit en page 103.)

Farid Katiraei, Reza Iravani, Nikos Hatziargyriou, and Aris Dimeas. Microgrids management. *Ieee Power And Energy Magazine*, 2008. (Cit en pages 127 et 128.)

Leonard Kaufman and Peter J. Rousseeuw. *Clustering Large Applications (Program CLARA)*, pages 126–163. John Wiley & Sons, Inc., 2008. (Cit en pages 66 et 76.)

Eric Keller and Jennifer Rexford. The "platform as a service" model for networking. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. USENIX Association, 2010. (Cit en page 114.)

J.O. Kephart. Research challenges of autonomic computing. *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 15–22, 2005. (Cit en pages 27, 28, 29 et 32.)

J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1) :41–50, January 2003. (Cit en pages 27, 28, 29 et 32.)

R.J. Kuo, Y.C. Wang, and F.C. Tien. Integration of artificial neural network and MADA methods for green supplier selection. *Journal of Cleaner Production*, 2010. (Cit en page 117.)

Gérard Le Lann. Distributed systems - towards a formal approach. In *IFIP Congress*, 1977. (Cit en page 57.)

Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998. (Cit en page 48.)

Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp : comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010. (Cit en page 116.)

J. Li, Massachusetts Institute of Technology. Department of Electrical Engineering, and Computer Science. *Agent Organization in the Knowledge Plane*. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2008. (Cit en pages 50, 73 et 137.)

Jian Liang, Rakesh Kumar, and Keith W Ross. Understanding kazaa. *Submetido para publicaçao*, 2004. (Cit en page 58.)

Virginia Lo, Dayi Zhou, Yuhong Liu, Chris Gauthierdickey, and Jun Li. Scalable supernode selection in peer-to-peer overlay networks. In *In Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems, La*. IEEE, 2005. (Cit en pages 55 et 56.)

Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-securing ad hoc wireless networks. In *ISCC '02 : Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, 2002. (Cit en page 30.)

Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane : an

information plane for distributed services. In *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006. (Cit en pages 46 et 47.)

Paul Marrow and Antonio Manzalini. The cascadas project : A vision of autonomic self-organizing component-ware for ict services. *ITSSA*, 2006. (Cit en pages 11, 37, 38, 49 et 137.)

K. McCloghrie and M. T. Rose. *Management Information Base for Network Management of TCP/IP-based internets :MIB-II*. RFC Editor, 1991. (Cit en pages 48 et 140.)

Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 2011. (Cit en pages 113 et 125.)

A. Mellouk, H. A. Tran, and S. Hoceini. *Quality-of-Experience for Multimedia*. Wiley-ISTE, 2013. (Cit en page 80.)

Peter Merz, Matthias Priebe, and Steffen Wolf. Super-peer selection in peer-to-peer networks using network coordinates. In *Internet and Web Applications and Services, 2008. ICIW'08. Third International Conference on*. IEEE, 2008. (Cit en pages 58 et 61.)

Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 2004. (Cit en page 105.)

Lionel Molinier, Mathieu Ligocki, Emna Ghedira, Guy Pujolle, and Dominique Gaïti. Piloting the spanning tree protocol in home networks using a multi-agent system. *Telecommunication Systems*, 2012. (Cit en page 30.)

Alberto Montresor. A robust protocol for building superpeer overlay topologies. In *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*. IEEE, 2004. (Cit en pages 58 et 60.)

K. Nagothu, B. Kelley, M. Jamshidi, and A. Rajaee. Persistent net-ami for microgrid infrastructure using cognitive radio on cloud data centers. *Systems Journal, IEEE*, 2012. (Cit en page 130.)

Jean-Pierre Nakache and Josiane Confais. *Approche pragmatique de la classification : arbres hiérarchiques, partitionnements*. Ed. Technip, Paris, 2004. (Cit en pages 63 et 64.)

Gérard Nguengang, Thomas Bullot, Dominique Gaiti, Louis Hugues, and Guy Pujolle. Autonomic resource regulation in ip military networks : A situatedness based knowledge plane. In *Advanced Autonomic Networking and Communication*. Springer, 2008. (Cit en pages 49, 54 et 137.)

Open-Networking-Foundation. Software-defined networking : The new norm for networks. Technical report, Open-Networking-Foundation, 2012. (Cit en page 27.)

ITUT Rec. P. 800.1,"mean opinion score (mos) terminology ". *International Telecommunication Union, Geneva*, 2006. (Cit en pages 46 et 97.)

Matei Ripeanu. Peer-to-peer architecture case study : Gnutella network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99–100. IEEE, 2001. (Cit en page 59.)

Diego Pinto Roa, Benjamín Barán, and Carlos A Brizuela. Wavelength converter allocation in optical networks : An evolutionary multi-objective optimization approach. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 414–419. IEEE, 2009. (Cit en page 101.)

Alvaro Rubio-Largo, Miguel A Vega-Rodríguez, Juan A Gómez-Pulido, and Juan M Sánchez-Pérez. Solving the routing and wavelength assignment problem in wdm networks by using a multiobjective variable neighborhood search algorithm. In *Soft computing models in industrial and environmental applications, 5th international workshop (SOCO 2010)*, pages 47–54. Springer, 2010a. (Cit en page 101.)

Alvaro Rubio-Largo, Miguel A Vega-Rodríguez, Juan A Gómez-Pulido, and Juan Manuel Sánchez-Pérez. A differential evolution with pareto tournaments for solving the routing and wavelength assignment problem in wdm networks. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010b. (Cit en page 100.)

J H Saltzer, D P Reed, and D D Clark. End-to-end arguments in system design. *Distributed Computing*, pages 509–512, 1991. (Cit en pages 21 et 39.)

S. Schuetz, K. Zimmermann, G. Nunzi, S. Schmid, and M. Brunner. Autonomic and decentralized management of wireless access networks. *Network and Service Management, IEEE Transactions on*, 2007. (Cit en pages 37, 49 et 137.)

Dennis Schwerdel, David Hock, Daniel Günther, Bernd Reuther, Paul Müller, and Phuoc Tran-Gia. Tomato-a network experimentation tool. In *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 1–10. Springer, 2012. (Cit en page 143.)

Raquel Sebastião and João Gama. A study on change detection methods. In *New Trends in Artificial Intelligence. 14th Portuguese Conference on Artificial Intelligence, EPIA*, October 2009. (Cit en pages 71 et 125.)

Neha Soni and Amit Ganatra. Comparative study of several clustering algorithms. *International Journal of Advanced Computer Research (IJACR)*, December 2012. (Cit en pages 63, 65 et 138.)

Roy Sterritt, Manish Parashar, Huaglory Tianfield, and Rainer Unland. A concise introduction to autonomic computing. *Adv. Eng. Inform.*, 2005. (Cit en page 28.)

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, 2001. (Cit en pages 49, 80, 92 et 139.)

John Strassner, N. Agoulmine, and E. Lehtihet. Focale : A novel autonomic networking architecture. 2006. (Cit en pages 37, 38, 50 et 137.)

Yongning Tang, Ehab Al-Shaer, and Bin Zhang 0007. Toward globally optimal event monitoring & aggregation for large-scale overlay networks. In *Integrated Network Management*, 2007. (Cit en pages 49 et 137.)

Hai Anh Tran, Abdelhamid Mellouk, Julien Perez, Said Hoceini, and Sherali Zeadally. Qoe-based server selection for content distribution networks. *IEEE Transactions on Computers*, 99, 2013. (Cit en page 101.)

András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Simutools '08 : Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. (Cit en page 90.)

D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems : Implications for the autonomous development of mental capabilities in computational agents. *Evolutionary Computation, IEEE Transactions on*, 2007. (Cit en page 43.)

Bernard M. Waxman. Broadband switching. chapter Routing of multipoint connections. 1991. (Cit en page 91.)

Steffen Wolf and Peter Merz. Evolutionary local search for the super-peer selection problem and the p -hub median problem. In Thomas Bartz-Beielstein, María J. Blesa Aguilera, Christian Blum, Boris Naujoks, Andrea Roli, Günter Rudolph, and Michael Sampels, editors, *Hybrid Metaheuristics*, Lecture Notes in Computer Science. Springer, 2007. (Cit en pages 58, 59, 60 et 88.)

Michael Wooldridge and Nicholas R. Jennings. Intelligent agents : Theory and practice. *Knowledge Engineering Review*, 1994. (Cit en page 31.)

Byunggu Yu, Alfredo Cuzzocrea, Dong Jeong, and Sergey Maydebura. On managing very large sensor-network data using bigtable. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, CCGRID '12, 2012. (Cit en pages 50 et 137.)

Résumé : La croissance du réseau Internet, l'émergence de nouveaux besoins par l'avènement des terminaux dits intelligents (smartphones, tablettes tactiles, etc.) et l'apparition de nouvelles applications sous-jacentes induisent de nombreuses mutations dans l'usage de plus en plus massif des technologies de l'information dans notre vie quotidienne et dans tous les secteurs d'activités. Ces nouveaux usages ont nécessité de repenser le fondement même de l'architecture réseau qui a eu pour conséquence l'émergence de nouveaux concepts basés sur une vue "centrée sur l'usage" en lieu et place d'une vue "centrée sur le réseau". De fait, les mécanismes de contrôle du réseau de transport doivent non seulement exploiter les informations relatives aux plans de données, de contrôle et de gestion, mais aussi les connaissances, acquises ou apprises par inférence déductive ou inductive, sur l'état courant du réseau (trafic, ressources, rendu de l'application, etc.) de manière à accélérer la prise de décision par les éléments de contrôle du réseau. Les travaux faits dans le cadre de cette thèse concernent ce dernier aspect et rejoignent plus généralement ceux tournés sur les réseaux autonomiques. Il s'agit dans cette thèse de mettre en oeuvre des méthodes relatives à la gestion, à la distribution et à l'exploitation des connaissances nécessaires au bon fonctionnement du réseau de transport. Le plan de connaissances mis en oeuvre ici se base à la fois sur l'idée de développer une gestion au sein d'une structure hiérarchisée et adaptative où seuls certains noeuds sélectionnés sont en charge de la dissémination des connaissances et l'idée de relier ces noeuds au travers d'un ensemble de réseaux couvrants spécialisés permettant de faciliter l'exploitation de ces connaissances. Comparée aux plateformes traditionnellement utilisées, celle développée dans le cadre de cette thèse montre clairement l'intérêt des algorithmes élaborés au regard des temps d'accès, de distribution et de partage de charge entre les noeuds de contrôle pour la gestion des connaissances. A des fins de validation, cette plateforme a été utilisée dans deux exemples d'application : le Cloud computing et les smartgrids.

Mots clés : Plan de connaissances, Qualité de Service, Approches Adaptatives, Réseaux Couvrants, Réseau Autonomique, Réseau Centré sur l'Information, Cloud Computing, SmartGrid

Abstract :

The growth of the Internet , the emergence of new needs expressed by the advent of smart devices (smartphones, touchpads , etc.) and the development of new underlying applications induce many changes in the use of information technology in our everyday life and in all sectors. This new use that match new needs required to rethink the foundation of the network architecture itself, which has resulted in the emergence of new concepts based on a “user-centeric” view instead of a “network-centric” view. In fact, the control mechanisms of the transmission network must not only exploit the information on data , control and management planes, but also the knowledge acquired or learned by inductive or deductive inference on the current state of the network (traffic, resources , the rendering of the application , etc.) to accelerate decision making by the control elements of the network. This thesis is dealing with this latter aspect, which makes it consistent with work done on autonomic networks. It is about conceiving and implementing methods for the management , distribution and exploitation of knowledge necessary for the proper functioning of of the transmission network. The knowledge plane that we implemented is based on both the idea of developing a management within an adaptive hierarchical structure where only some selected nodes are responsible for the dissemination of knowledge and the idea of linking these nodes through a spanning set of specialized networks to facilitate the exploitation of this knowledge. Compared to traditionally used platforms , the one developed in this thesis clearly shows the interest of the developed algorithms in terms of access time , distribution and load sharing between the control nodes for knowledge management. For validation purposes , our platform was tested on two application examples : Cloud computing and smart grids.

Keywords : Knowledge plane, Quality of Service, Adaptive approaches, spanning networks, Autonomic Networks, Information-centric network, Cloud Computing, SmartGrid